Západočeská univerzita v Plzni

Detekce řeči v rozhlasových nahrávkách

Katedra kybernetiky

Diplomová práce

Autor:
Luděk Müller

FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Luděk MÜLLER**
Osobní číslo:	**A20N0029P**
Studijní program:	**N3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Téma práce:	**Detekce hudby a řeči v rozhlasových nahrávkách**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Prostudujte možnosti detekce hudby a řeči v audionahrávkách metodami založenými na hlubokých neuronových sítích.
2. Navrhněte detektor vhodný pro detekci hudby a řeči v rozhlasových nahrávkách.
3. Experimentálně ověřte navržené metody na reálných datech.

Rozsah diplomové práce: **40 – 50 stránek A4**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná**

Seznam doporučené literatury:

1) Pecinovský, Rudolf, Python : kompletní příručka jazyka pro verzi 3.9, Grada Publishing, Praha 2020
2) Chollet, François, překlad: Rudolf Pecinovský, Deep learning v jazyku Python : knihovny Keras, Tensorflow, Grada Publishing, Prah, 2019
3) Goodfellow, Ian, Yoshua Bengio and Aaron Courville, Deep learning / Adaptive computation and machine learning, Cambridge, Massachusetts, 2016
4) Kamath, Uday, Liu, John, Whitaker, James Deep learning for NLP and speech recognition, Springer, Cham Švýcarsko, 2019
5) Doukhan, David and Carrive, Jean and Vallet, Félicien and Larcher, Anthony and Meignier, Sylvain, „An Open-Source Speaker Gender Detection Framework for Monitoring Gender Equality", Acoustics Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on 2018

Vedoucí diplomové práce: **Ing. Jan Zelinka, PhD.**
Katedra kybernetiky

Datum zadání diplomové práce: **1. října 2021**
Termín odevzdání diplomové práce: **23. května 2022**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 1. října 2021

P R O H L Á Š E N Í

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni. Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 26.08.2022 _____

Luděk Müller

Poděkování

Rád bych poděkoval vedoucímu své diplomové práce Ing. Janu Zelinkovi, Ph.D. za jeho odborné vedení, cenné připomínky při sepisování této práce i ochotu, s jakou mi pomohl dokončit tuto práci.

Také děkuji Ing. Janu Lehečkovi, Ph.D. za jeho rady poskytnuté modely wav2vec i návody k jejich využití.

Na závěr bych rád poděkoval mé rodině za velkou podporu.

Abstrakt

Cílem této práce je navrhnout a experimentálně odzkoušet novou metodu detekce řečových a neřečových částí rozhlasového vysílání, jež by měla přispět k úspěšnému vyřešení úlohy automatického určení začátků a konců rozhlasových pořadů za účelem jejich bezproblémového vystavování na internetovém portále mujRozhlas.cz.

Je prezentováno stávající řešení, které je v současné době používané. Dále jsou v práci popsány nejvíce slibné metody založené na konvolučních neuronových sítích a state-of-the-art metodách využívajících transformery a wav2vec framework.

Výsledkem práce je pak kromě vytvoření sady datasetů a skriptů pro porovnání jednotlivých metod především navržení a realizace nového detektoru splňujícího podmínky kladené na jeho implementaci pro řešení úlohy automatické detekce začátků a konců pořadů v Českém rozhlase.

Abstract

This thesis aims to design and experimentally test a new optimal method of a voice-activity-detector, which should add successfully solve the task of automatically determining the beginnings and ends of radio broadcasts programs for the purpose of their seamless display on the internet portal mujRozhlas.cz

An existing solution that is currently in use is presented. The work also describes the most promising methods based on convolutional neural networks and state-of-the-art methods using transformers and the wav2vec framework.

The primary result of this work is the design of a new detector meeting the conditions set for its implementation to solve the task of automatic detection of the beginnings and ends of programs on the Czech Radio, as well as the creation of a set of datasets and scripts for comparing individual methods.

Obsah

1 Úvod

Tato diplomová práce si klade za cíl přispět k úspěšné realizaci projektu mujRozhlas, jehož nedílnou součástí je úloha automatického vystavení vybraných pořadů Českého rozhlasu na internetovém portálu Českého rozhlasu *mujRozhlas.cz*. Aby se daný rozhlasový pořad mohl na portále vystavit, musí být správně nalezen jeho začátek a konec. Porušení této podmínky vede buď k nežádoucímu oříznutí (zkrácení) pořadu, a nebo v horším případě, i k právním sporům, pokud by na portále vystavený úsek audia obsahoval část okolního pořadu, na který se vztahují licenční podmínky, jako je tomu např. u hudební skladby. Diplomová práce řeší jeden z dílčích úkolů této úlohy a to konkrétně detekci audio úseků, obsahujících mluvenou řeč, tedy rozlišení časových úseků, kde je v akustickém signálu mluvené slovo a kde tomu tak není. V této práci je popsán způsob již nevyhovujícího stávajícího řešení, dále se popisují návrhy nových možných způsobů řešení, které by mohly přinést zlepšení v oblasti výpočetní náročnosti nebo v přesnosti samotného rozhodování. Důležitou a nezanedbatelnou částí je pak detailní popis přípravy datasetů pro trénování, validaci a nastavení optimálních hyperparametrů použitých modelů. Z implementačního hlediska, je třeba uvést, že metody budou spouštěné v praxi na serverech Českého rozhlasu, s Intel(R) Core(TM) i7-10750H CPU, bez možnosti výpočtu na grafických kartách. Tato skutečnost použití konkrétního HW, na kterém bude výsledný program implementován a spouštěn, vytváří omezující podmínku na náročnost výpočtu. Pro implementační fázi je požadováno, aby celková výpočetní náročnost detekce nepřesáhla více jak o 20 % výpočetní náročnosti stávajícího řešení.

2 Motivace a cíle práce

Největší motivací této práce je možnost podílet se na praktické aplikaci přesného rozpoznávání částí audiosignálu obsahujících mluvené slovo a tak pomoci přesně identifikovat začátky a konce nahrávek audio pořadů. Cílem práce je nejprve vyhodnotit účinnost a přesnost stávajícího řešení, dále nalézt vhodné nové metody řešení, které se zabývají podobným problémem - detekcí řeči v akustickém signálu. Tyto metody pak následně upravit do podoby možného nasazení do takového režimu, který by vyhovoval omezujícím podmínkám zadané úlohy. Posléze všechny tyto metody porovnat, najít optimální kritéria, jak metody správně vyhodnotit a určit přesnost a požadavky na výpočetní výkon. Český rozhlas disponuje značně rozsáhlým on-line archivem, který obsahuje jak uložená vysílání, tak i k těmto audio datům přiřazená metadata, tedy datasety obsahující podpůrné informace, jako např. jména moderátorů, popisy stanic a vysílaných programů, délky, či místa, kde byl pořad vysílán. Tato metadata by byla velkým přínosem pro správné určení detekce řečových úseků (zejména, pokud by metadata obsahovala údaje o času, kdy je pořad vysílán, jaký jiný pořad mu předchází a jaký po něm následuje, či informaci o typu pořadu, např. jedná-li se o bohoslužby či operní zpěv, což by mělo za následek on-line adaptaci parametrů rozpoznávače, čímž by se např. mohlo dosáhnout lepší přesnosti detekce), avšak bohužel jsou tato metadata mnohdy nepřesná a v mnoha případech obsahují nesrovnalosti či nejsou k dispozici. U dříve vysílaných pořadů jsou metadata v mnoha rozdílných formátech, nebo dokonce zcela chybí. Motivací této práce je pomoci tento on-line archiv kontinuálního vysílání analyzovat a přesně zpracovat tak, aby uživatelé internetu mohli přistupovat ke konkrétním pořadům či jejich úsekům dle přidělených práv. Smyslem celého projektu mujRozhlas je vytvořit systém, který by měl jak archiv historických, tak i živých vysílání (zde je podmínka zpracovávat vysílání v reálném čase) automaticky rozšiřovat o audiodata a on-line zpřístupňovat vybrané pořady potenciálním zájemcům. Analýza vysílání je pouze první polovinou celého projektu. Druhou částí je Analýza mluveného slova, tedy převod hlasu na text s rozlišením mluvčích či jejich pohlaví a případně identifikaci mluvčího. Druhá část je řešena samostatným projektem. Celkovým

výsledkem, ve kterém dochází k propojení obou částí, je podrobný popis zvukové stopy včetně převodu hlasu na text a to všech stanic Českého rozhlasu. A tento výstup, tedy archivy přímo ze živého vysílání, by měl být zpřístupněn uživatelům na internetu dle licenčních podmínek v rámci projektu mujRozhlas. Projekt mujRozhlas je strategickým krokem Českého rozhlasu a bude největším českým audio archivem na internetu. Analytika vysílání je založená na vývoji doposud neexistujícího softwaru (SW). Cílem této práce je podpořit vývoj tohoto SW. Pro dosažení tohoto cíle jsem nadefinoval množinu následujících podcílů:

1. Analyzovat metody možného řešení.
2. Porovnat je se stávajícím řešením i porovnat je mezi sebou navzájem.
3. Vytvořit datové sady pro porovnávání různých metod i pro vyhodnocení nového možného řešení.
4. Na základě provedené analýzy a porovnání jednotlivých metod navrhnout řešení ideálně vedoucí ke zlepšení stávajícího stavu.
5. Experimentálně ověřit úspěšnost navrženého řešení.
6. Vytvořit SW balíček, vhodný pro implementaci v Českém rozhlase.

3 Stávající řešení

Dosavadní řešení detekce řečového signálu nasazené a implementované v Český rozhlas je založené na neuronových konvolučních sítích. Konvoluční neuronová síť (CNN), je speciální třída neuronových sítí, která je vhodná na zpracování dat vyznačujících se určitou topologii, jakou má např. digitální obraz v podobě 2D mřížky, jež obsahuje hodnoty jasu a barvy (RGB) každého pixelu obrazu. Podobně jako mřížka v digitálním obraze, se i posloupnost audiovzorků může zpracovávat jako mřížka, ale konvoluční okénko, se kterým síť počítá, je na rozdíl od případu digitálního obrazu jednorozměrné.

3.1 Konvoluční neuronová síť

Podobně jako je tomu při zpracování obrazu, je při zpracování zvuku zpracováváno obrovské množství informací. Architektura CNN vychází z principu lidského vnímání, tedy že při zpracování obrazu člověkem, každý bio neuron reaguje na podněty pouze v omezené oblasti zorného pole. V systému biologického vidění se část tohoto pole nazývá receptivní pole neuronu. Jaždý neuron v CNN tedy zpracovává data pouze ve svém receptivním poli. Každý neuron je pak spojen s ostatními neurony tak, že prakticky pokrývají celé zorné pole. Vrstvy CNN jsou uspořádány takovým způsobem, že nejprve detekují jednodušší vzory (např. v případě počítačového vidění to mohou být čáry, křivky atd.) a dále složitější vzory (obličeje, objekty atd.). Pomocí CNN lze interpretovat zpracování dat počítačem analogicky k člověku.

Konvoluční vrstva je základním stavebním kamenem CNN. Nese hlavní část výpočetní zátěže sítě. Tato vrstva provádí „skalární “ součin dvou matic (součet všech součinů prvku jedné matice s prvkem stejné pozice matice druhé), kde jedna matice je množina naučitelných parametrů jinak známých jako jádro a druhá matice je omezená část receptivního pole. Jádro je v prvních dvou dimenzích menší než obrázek, ale může mít třetí rozměr - hloubku. To znamená, že pokud je obraz složen ze tří (RGB) kanálů, výška a šířka jádra budou „v rovině obrázku“ malé, ale hloubka je rovna počtu kanálů.

Během dopředného průchodu se jádro posouvá po výšce a šířce obrazu a vytváří obrazovou reprezentaci této receptivní oblasti. To vytváří dvourozměr-

nou reprezentaci obrazu známou jako aktivační mapa, která poskytuje odezvu jádra v každé prostorové poloze obrazu. Posuvná velikost jádra se nazývá krok (stride).

V případě audiosignálu je třeba uvažovat místo 2D obrázku 1D posloupnost zvukových vzorků a místo jádra 2D (s třetím rozměrem hloubkou) jádro 1D (s hloubkou).

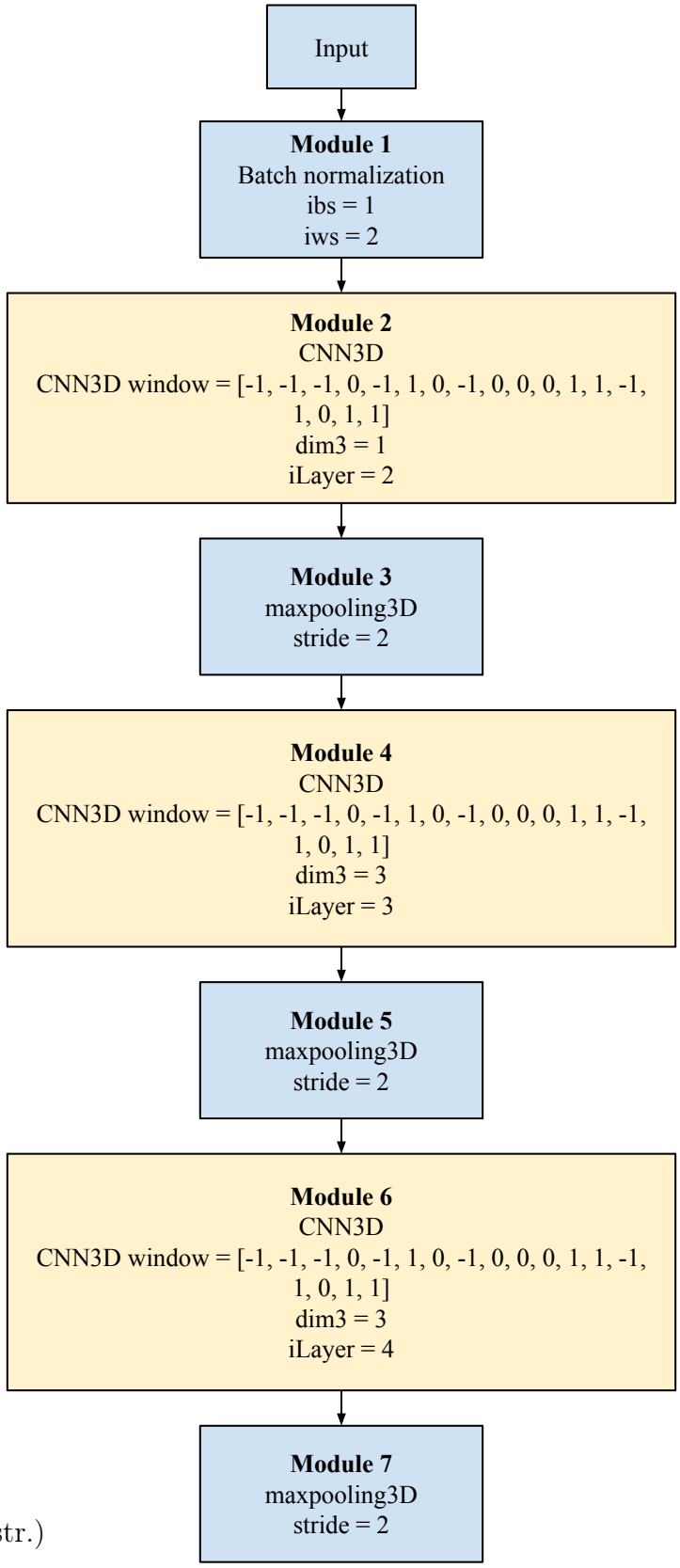
3.2 Architektura užité CNN

Stávající řešení na serverech ČRo je založená právě na CNN. Vstupem je audiosignál a výstupem této CNN je pro každých 10ms vektor tří hodnot (speech, music, music+speech). Datový typ je float vyjadřující úroveň dané hodnoty.

Hodnoty vycházející z CNN nejsou vyhlazovány s ohledem na hodnoty okolí. Často tedy nastává poměrně bohaté přeskakování kategorií na krátkém časovém intervalu. Výstupy z CNN jsou proto dále heuristickým algoritmem založeném na počtu rozhodnutí náležejících do stejné třídy (tj. např. náležejících do třídy řeč) v určitém časovém (a v čase klouzajícím) okénku upraveny tak, aby byly vyhlazeny.

Celý algoritmus není nutné do detailu popisovat. Neuvedení jeho přesného popisu totiž není překážkou k dosažení cílů této práce. Pokud nově navržený způsob řešení dosáhne ve srovnání se stávajícím řešením statisticky významně lepších výsledků, bude stávající řešení nahrazeno novým. Důležité tedy bude jen mít možnost provést relativně objektivní srovnání stávajícího řešení z hlediska přesnosti a náročnosti s řešením novým.

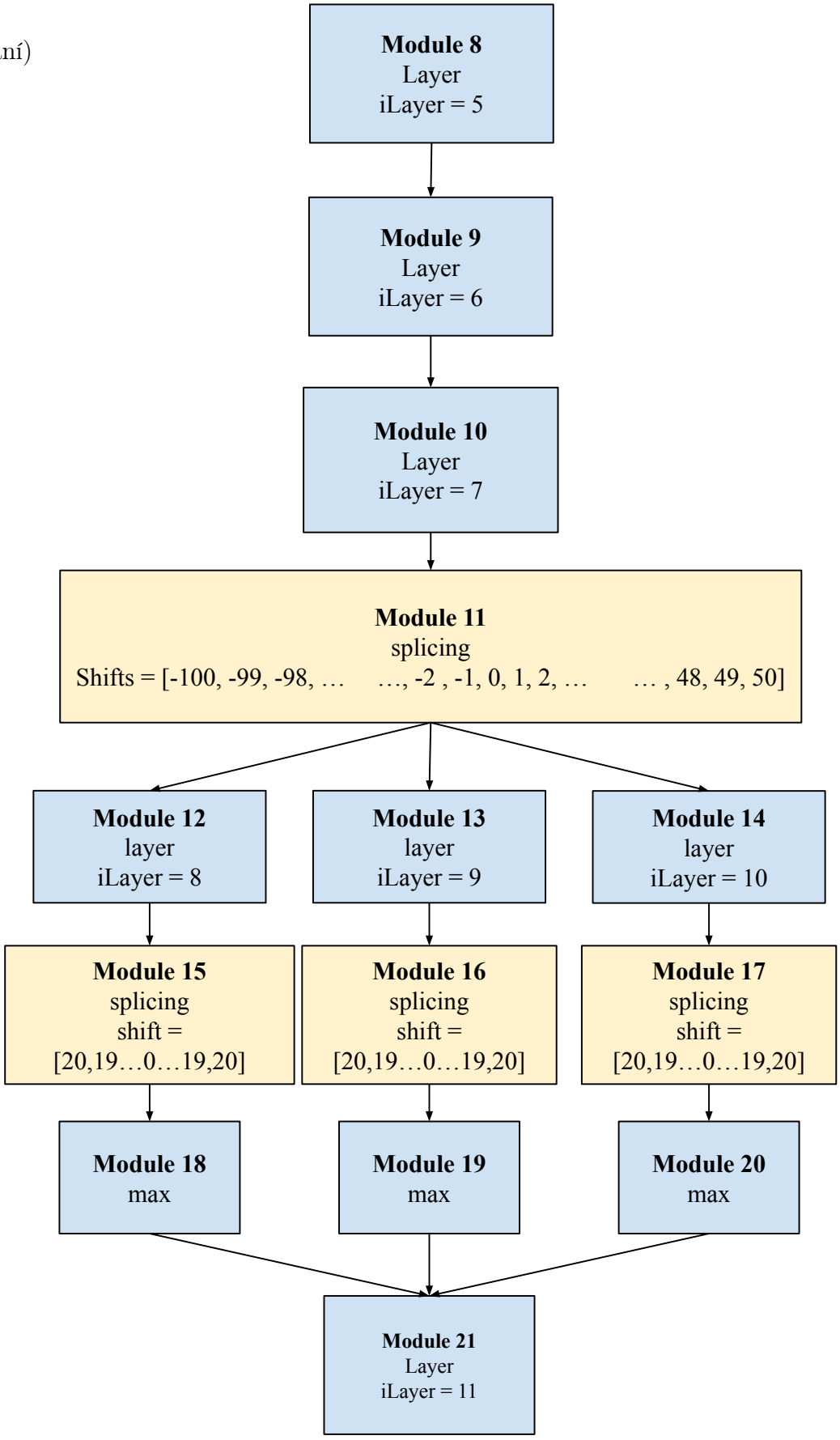
Na následující stránce je na obr. 1 uvedeno schéma CNN sítě stávajícího řešení:



(pokračování na další str.)

Obrázek 1: popis schéma stávajícího řešení, první část

(pokračování)



Obrázek 2: popis schéma stavájícího řešení druhá část

Vstupem sítě je navzorkovaný audiosignál s 16 kHz vzorkovací frekvencí. První vrstva řeší "batch"normalizaci, která normalizuje vstup do dalších vrstev pro každou dávku vstupních dat. To má za následek stabilizaci procesu učení a dramatické snížení počtu tréninkových epoch potřebných k trénování hlubokých sítí.

Pro batch normalizaci se počítají střední hodnoty a směrodatné odchylky dané dávky („batchi“) a jejich aplikací na každou vstupní proměnnou a cílem obdržet nová normalizovaná data dávky.

[ibs = 1] značí identifikátor vektoru záporů biasů, to je střední hodnota, kterou se normalizuje,

[iws = 2] pak značí identifikátor vektoru vah, tj inverzních hodnot směrodatných odchylek ($1/\sigma$).

Pro sdružování jader se využívá metodu maxpooling [pooling, how = max].

Zkratka [dim3] značí počet jader konvoluční sítě.

Splicing značí spojení vrstev dle konkrétního shiftu

Shiftem dochází k posunu prvků vstupního vektoru o předepsaný posun, kde prvky jsou posunuty kladně (směrem k větším indexům) o offset posunu podél rozměru osy. Záporné hodnoty posunu posunou prvky v opačném směru. Prvky, které přešly přes poslední pozici, se zapíší od první pozice dále a naopak.

V poslední vrstvě pak dochází ke zřetězení vektorů.

4 Analýza problému

Stávající řešení automatické detekce začátků a konců vybraných pořadů Českého rozhlasu trpí určitými nedostatky, především v případech, kdy nejsou k dispozici přesná metadata o složení vysílání, kdy dochází k nedodržení časového rozvrhu plánu jednotlivých pořadů nebo když metadata zcela chybí. Častým případem je vysílání z tzv. „konzerv“, kde „konzerva“ je kontinuální záznam v minulosti vysílané posloupnosti pořadů bez informace o vnitřních časech a složení „konzervy“. Metadata sice mohou obsahovat údaje o začátku a konci „konzervy“, ale již nenesou žádnou informaci o vnitřním obsahu „konzervy“, tedy např. o tom, že „konzerva“ obsahuje jednu nebo více písní, reklam apod.

Velkým problémem je též současný trend ve vysílání Českého rozhlasu spočívající ve snaze neznatelně a plynule přecházet mezi koncem jednoho a začátkem následujícího pořadu. V takových případech například moderátor před koncem předcházejícího pořadu začne hovořit o pořadu následujícím, aniž by z jeho projevu bylo pro posluchače patrné, že se obsah hovoru moderátora váže již k dalšímu pořadu. Například před začátkem vlastního pořadu se může ve vysílání objevit (úryvek) skladby, nebo jiného vystoupení hosta následujícího pořadu. Poté následuje úvodní znělka začátku pořadu a po ní (nebo již během ní) moderátor uvádí další pořad s tím, že předchozí ukázka se váže k hostu současného pořadu. Český rozhlas pak požaduje, aby mohl na svém internetovém portálu vystavit nejen část pořadu od úvodní znělky dále, ale též pořadu předcházející ukázku jako součást celého pořadu. V častých případech pak znělky nezačínají stejně, netrvají stejnou pravidelnou dobu, jsou překryté okolním vysíláním, ať již hlasem moderátora či okolní hudbou, mohou se měnit a velmi často nejsou ve vysílání obsaženy vůbec.

Často ani pracovníci Českého rozhlasu nedokáží určit, kdy který pořad přesně začíná a kdy končí a rozhodnout musí specialista (většinou pracovníci zodpovědní za výrobu a vysílání konkrétního pořadu). Dost často je ke správné detekci hranic pořadů potřebné i porozumění obsahu vysílání, např. když se moderátor před koncem pořadu loučí, nebo naopak na začátku dalšího pořadu bez rozdělující události mezi pořady (např. znělky) oznamuje, že předchozí

pořad skončil. Podobně i před začátkem pořadu může posluchače informovat, že následující pořad začne nebo již začal apod.

Původním záměrem Českého rozhlasu bylo pro potřeby automatického vystavování rozhlasových pořadů na internetovém portálu mujRozhlas.cz vytvořit automat postavený na metodách strojového učení a umělé inteligenci. Z výše uvedeného textu vyplývá, že tento záměr je v současnosti nerealizovatelný, neboť ani člověk – pracovník rozhlasu - nezvládá tuto úlohu, neexistují referenční data (informace od učitele) s přesnými údaji o začátku a konci pořadu, a pokud nějaká (ne zcela spolehlivá) data jsou, jsou vzhledem k obrovské variabilitě událostí vyskytujících se na přechodu pořadů vysílání a velkého množství řídkých výjimek, v cela nedostatečném množství. Nelze se spolehlivě opřít ani o údaje z metadat. Řešení úlohy se tedy opírá o znalosti specialistů – expertů, kteří dokáží předat znalosti potřebné ke správnému řešení popsané úlohy. Zde je třeba poznamenat, že množina potřebných znalostí se může lišit pořad od pořadu.

Z uvedeného vyplývá, že řešení úlohy by se mělo opírat spíše o implementaci znalostního či ještě přesněji expertního systému, který by byl vybaven potřebnou množinou znalostí ve své znalostní bázi. Touto cestou se praktická implementace řešení úlohy detekce začátků a konců jednotlivých pořadů skutečně ubírá, kdy ke každému pořadu je přiřazen tzv. profil pořadu, ve kterém jsou mimo jiné obsažena pravidla, podle kterých se systém detekce hranic pořadů rozhoduje.

Pravidla obsažená v profilech pořadů se velmi často opírají o informaci, zda daný časový úsek rozhlasového vysílání nese informaci v řečové podobě, nebo se o řeč nejedná (např. v případě hudby, znělky, reklamy - zde je třeba navíc umět zjistit, zda případně se vyskytující řeč je součást reklamy, různých neřečových jevů jako je řev motoru, ale i potlesk, smích apod.). Důležitou podporou pro určení, zda se v daném časovém úseku vysílání objevuje řeč, či jiný neřečový projev, tedy musí být systém detekce řečového projevu během vysílání a odlišení těchto míst od úseků, kdy se o řeč nejedná.

Stávající řešení detekce řečových a neřečových úseků je založeno na CNN, které dosahují v mnoha podobných úlohách mnohem lepších výsledků, než

metody založené na parametrických statistických modelech, jako jsou např. HMM/GMM (Hidden Markov Model / Gaussian Mixture Model).

I po nasazení detektorů založených na neuronových sítích však detekce řeč/neřeč velmi často selhávala. Přitom správná detekce těchto úseků a jejich hranic je jedním z klíčových požadavků a nutných podmínek na správnou funkčnost celého systému. Z těchto důvodů bylo učiněno rozhodnutí, pokusit se zlepšit práci tohoto systému detekce řečových a neřečových úseků v proudu rozhlasového vysílání. Pro vyřešení tohoto úkolu bylo nejprve potřeba provést analýzu možných přístupů k řešení, vybrat vhodné kandidáty a provést jejich srovnávací testy i porovnání se stávajícím řešením (metodu stávajícího řešení budu dále značit zápisem ORG).

Důležitým požadavkem na detektor je kromě dosažení nejvyšší úspěšnosti i rychlost jeho práce v podmínkách Českého rozhlasu (např. nemožnost využití grafických karet GPU, paralelní zpracování desítek vysílacích kanálů atd.).

Při návrhu vhodného řešení se tedy sledují následující cíle:

1. Co největší přesnost.

2. Rychlost srovnatelná s původním řešením (nový detektor by měl pracovat rychleji než 1,2násobek průměrného výpočetního času stávající implementace).

3. Latence do 30 sec. Přestože se vybrané pořady nevystavují na internetový portál hned, je vhodné je zpracovat co nejrychleji, neboť pořad „vysekнutý“ z vysílání může procházet ještě dalším zpracováním a (lidskou) kontrolou, popřípadě opravou.

Dalším praktickým požadavkem bylo, aby detektor preferoval malou chybu prvního druhu, tj. chybu svého tvrzení, že nedetekuje žádnou řeč (přestože se řeč ve skutečnosti ve vstupním audiosignálu vyskytuje), před dosažením nízké chyby druhého druhu, tj. chyby svého tvrzení, že daný úsek audiosignálu je řeč, přestože ve skutečnosti řeč ve vyšetřovaném úseku audiosignálu není.

4.1 Analýza možných řešení

Různých návrhů řešení existuje již několik, rozpoznávání řeči je ve světě velmi rozšířené a má velký význam. Již od 80. let minulého století se vyvíjejí různé metody na rozpoznání mluveného slova. Detekce míst, kde je řeč vůbec vyslovována, pak většinou sloužila jako předzpracování samotným rozpoznávačům mluvené řeči. Do příchodu metod založených na principu neuronových sítí dominovaly ve vývoji zejména stochastické pravděpodobnostní modely založené na algoritmickém předzpracování zvukového signálu - kódování signálu nejprve na příznakové vektory (např. MFCC, PLP), které pak následně byly zpracovány akustickým modelem, jazykovým modelem a dekódovány na výslednou posloupnost slov, přičemž se využívaly trifony a difony, filtrace, okénkování a další metody založené na znalostech odborníků na zpracování signálu a statistického a jazykového modelování. Podobný přístup byl aplikován i u detektorů řeči a "neřeči". S příchodem neuronových sítí v r. 2014 ve však postupně tyto metody přestaly používat a byly nahrazovány stále propracovanějšími metodami založenými na metodách hlubokých neuronových sítích. Jedním z prvních, který přinesl výrazné zlepšení, pak byl inaSpeechSegmenter.

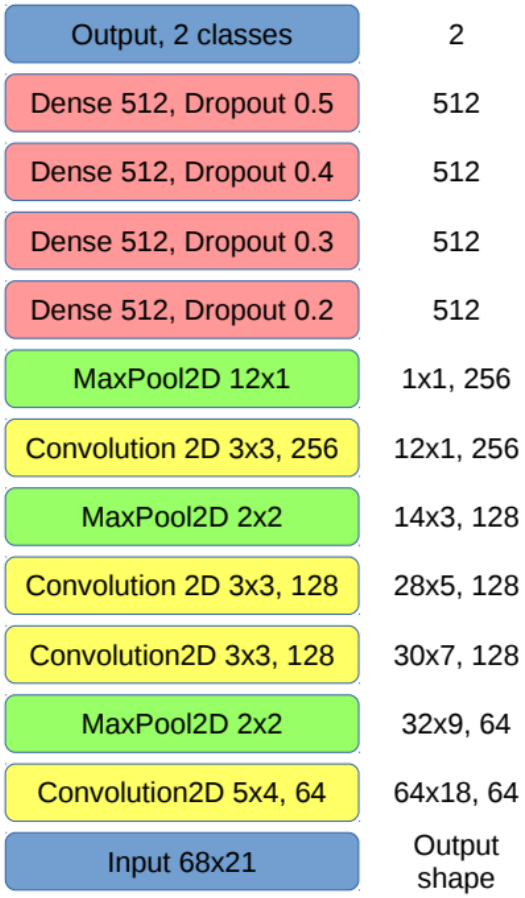
4.2 NINA

Jako první metoda, která se nabízí k možnému řešení našeho problému je inaSpeechSegmenter (dále také jako NINA), který v mezinárodní soutěži MIREX 2018, v oblasti detekce řeči/neřeči, získal v roce 2018 první místo [5]. Konkrétně v této práci jsem použil verzi inaSpeechSegmenter 0.7.3 pro porovnávací a testovací potřeby. Program inaSpeechSegmenter byl prezentován v roce 2018 v Kanadském Calgary na konferenci IEEE (International Conference on Acoustics, Speech and Signal Processing, ICASSP). InaSpeechSegmenter klasifikuje zvukovou stopu (tedy akustický signál) do tří homogenních zón: i) do oblasti, kde je detekována řeč, ii) oblasti kde je detekována hudba a iii) oblastí, kde je detekován šum, hluk a jiné zvukové efekty. Tento program byl také tvořen pro detekci pohlaví mluvčího. V zásadě inaSpeechSegmenter byl navržen za účelem provádění rozsáhlých studií genderové rovnosti založených na odhadu procenta doby mluvy řeči mužů a žen v médiích. Modely jsou optima-

lizovány pro francouzský jazyk, protože byly trénovány z nahrávek francouzsky mluvících osob (datasety pocházející z televizních a rozhlasových pořadů byly doplňovány o informaci identifikace mužského či ženského řečníka).

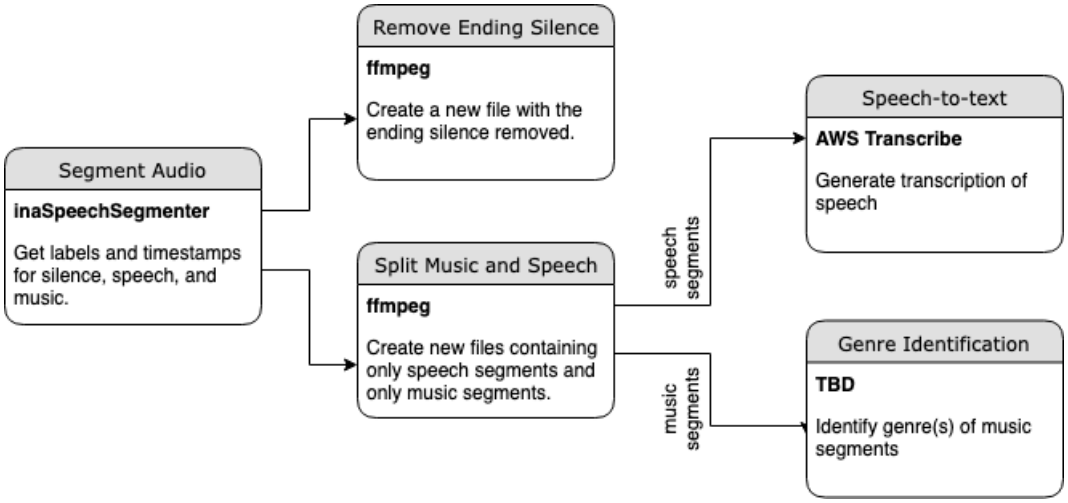
Doba mluvení daného pohlaví je popsána pomocí procenta doby řeči mluvy žen (WSTP, z angl. Women Speaking Time Percentage), která se odhaduje pomocí algoritmů automatické detekce pohlaví mluvčích. WSTP je použito napříč kanály, roky, hodinami a regiony. V našem případě však tato znalost nepřináší žádnou přidanou hodnotu a proto jsem úlohu rozpoznání pohlaví řečníka nezahrnul do cílů práce a metodu rozpoznání pohlaví deaktivoval z důvodu možného zrychlení zpracování počítačem.

InaSpeechSegmenter je systém postavený na předzpracování audiosignálu log-mel filterbankami (filtry na koncepci logaritmicko-melovských koeficientů) [6]. Model detektoru je postaven na 4 skrytých a 4 konvolučních vrstvách. Jak již bylo řečeno, natrénovaná CNN klasifikuje vstupní úsek audia do tříd řeči, hudby a úseků, které nejsou ani hudbou ani řečí. Dále zóny řeči mohou být rozděleny do menších segmentů označených podle pohlaví mluvčího (muž nebo žena). Oblasti řeči s hudbou, nebo řeči s hlukem jsou označeny jako řeč [5]. NINA podporuje všechny formáty medií akceptované programem FFmpeg (wav, mp3, mp4 atd.). Architektura sítě NINA je zobrazena na obr. 3.



Obrázek 3: popis schéma vrstev modelu NINA, obr. převzat [5]

Segmentace audia systému NINA se nazývá MGM (Metadata Generation Mechanismus). Výsledek MGM využili autoři NINA pro určení, jak velká část francouzského archivu má jaký obsah (např. jak velká část zabírá ticho). Segmentace též slouží k určení, kam daný segment audia poslat - systém může audio poslat na vstup systému STT (Speech-To-Text) nebo do systému vyhodnocení hudby.



Obrázek 4: fáze NINA, obr. převzat z [4]

Algoritmy strojového učení vyžadují trénovací data odpovídající kategoriím, které se mají naučit. Trénovací dataset pro učení CNN NINY se skládal z několika datasetů: GTZAN [11], Scheirer-Slaney [9] a MUSAN [10]. Dataset byl částečně automaticky zpracován, nicméně z poměrně větší části byl ručně anotován anotátory. Anotace audia byla realizována za použití poloautomatických anotačních postupů založených na optickém rozpoznávání znaků na obrazovce. Úryvky televizních zpráv se jménem osobnosti na obrazovce byly prezentovány anotátorům odpovědným za ruční ověřování.

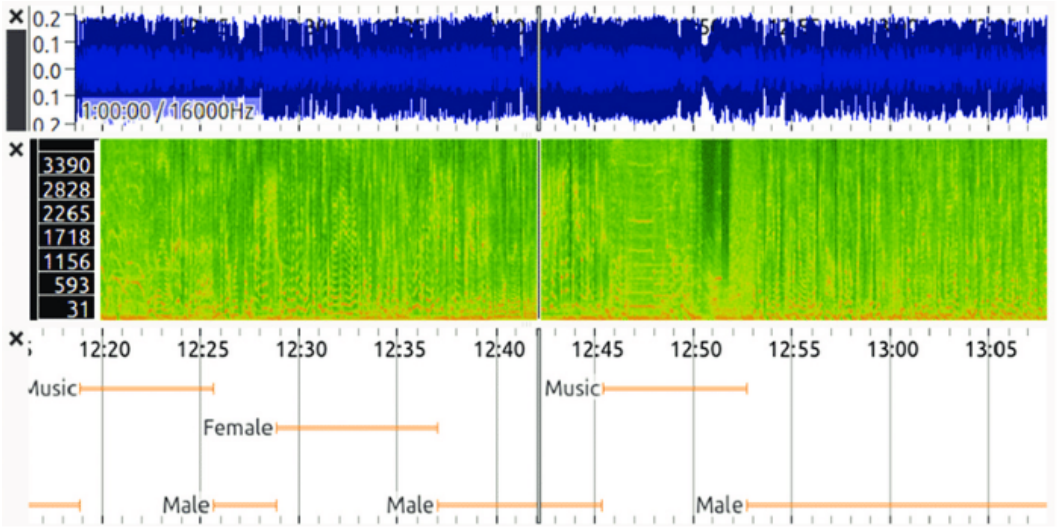
Data pocházela od francouzských televizních stanic, ale i rádiových nahrávek. Převážně se jednalo o vysílací časy denních hodin vysílaných v letech 2001 až 2018 na 22 televizních kanálech a 21 rozhlasových stanicích. Podle autorů tento trénovací dataset byl v době svého vzniku největší ručně anotovanou databází televizního a rozhlasového vysílání [3].

Modely detekce pohlaví byly trénovány pomocí slovníku mluvčích. Snahou bylo mít trénovací data dostatečně reprezentativní vzhledem k rozmanitosti zpracovávaných materiálů obsahujících různé přízvuky, styly mluvení, expre-

sivní modalitu, podmínky nahrávání atd. Výsledný slovník mluvčích byl vytvořen z nahrávek shromážděných od roku 1957 do roku 2012, což umožnilo komplexní reprezentaci stylů mluvení a podmínek nahrávání napříč desetiletími. Výsledný slovník obsahuje 32 000 vzorků řeči mluvčích, což odpovídá 1780 odlišným mužům (94 hodin) a 494 ženám (27 hodin).

Bohužel takto rozsáhlý anotovaný dataset nemáme k dispozici a již vůbec ne pro češtinu.

Pro zajímavost uvedu výsledky ohledně genderové rovnosti: Hodnocení InaSpeechSegmenter bylo provedeno podle jeho schopnosti odhadnout hodnotu poměru WSTP. Ukázalo se, že robustnost estimátoru WSTP je úměrná velikosti zpracovávaných dat, protože okamžité chyby detekce pohlaví se navzájem vyvažovaly při rozumně dlouhých časových intervalech. Hodnocení prováděná na ručně komentovaných televizních zprávách vedla k chybám odhadu WSTP pod 0,6% pro nahrávky delší než 30 minut.



Obrázek 5: ukázka interaktivního zobrazení výstupu automatické segmentace řeči. První vrstva je surový audio signál, druhá vrstva je časová frekvence reprezentace signálu. Poslední vrstvou je automatická predikce navrhovaného segmentátoru řeči v hudebním, mužském a ženském úryvku, obr. převzat z [4]

Výsledky ukázaly, že muži mluvili v televizi a rádiu v roce 2018 dvakrát více než ženy a že v roce 2004 mluvili třikrát déle než ženy. Také ukázaly, že pouze jedna rozhlasová stanice ze 43 uvažovaných kanálů dosáhla hodnoty WSTP větší než 50 %. Zjistilo se též, že WSTP je nižší během časových slotů s vysokou návštěvností na soukromých kanálech. Celkově projekt NINY před-

stavuje rozsáhlou studii genderové rovnosti založenou na automatické analýze audiovizuálního materiálu a dle autorů nabízí konkrétní perspektivy pro monitorování genderové rovnosti v médiích. Software použitý pro analýzu byl vydán jako open-source (bez anotovaných dat).

4.3 Wav2vec 2.0 modely

Architektura wav2vec 2.0 modelů vychází z architektury transformerů, proto nejdříve uvedu popis architektury transformerů:

4.3.1 Transformery

V roce 2017 byl publikován článek *Attention Is All You Need* [12], který závratně změnil používání metod založených na neuronových sítích a nastavil nový směr ubírání se vývoje strojového učení. Samotný model transformeru je založen právě na attention mechanismu. Nejznámější současné modely, které se objevují v úlohách NLP, se skládají z desítek transformerů nebo některých jejich variant, příkladem je GPT-2 nebo BERT. Stručně lze popsat transformery jako neuronové sítě spadající do oblasti hlubokého učení, které mají specifickou strukturu, která jim umožňuje se naučit, jak velkou pozornost mají věnovat jevům vyskytujícím se v okolním relativně širokém kontextu.

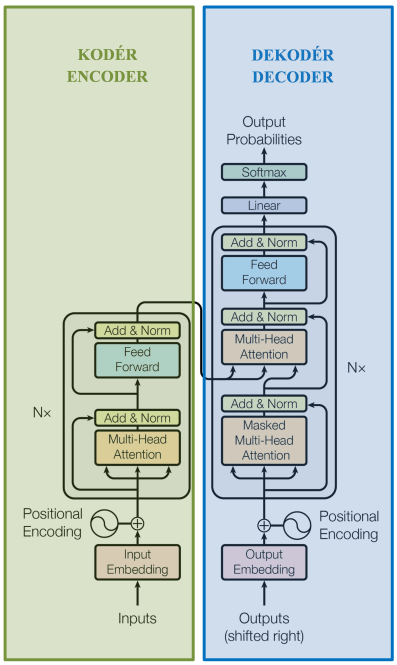
V "předtransformerové" době se strojové překladače založené na RNN (Rekurent Neural Network) vyznačovaly omezením při práci s dlouhými vstupními sekvencemi symbolů (slov). S rostoucím kontextem (historií) se ztrácela jejich schopnost uchovávat informace z dříve viděných dat. Při učení RNN sítí dochází k problému tzv. *vanishing gradientu*, tedy vymizení gradientu v důsledku dlouhého kontextu. Problém vymizení gradientu se snaží řešit modely LSTM (Long-Short-Term-Memory) nebo GRU (Gated Reccurent Unit) modely, ale výrazné zlepšení v úlohách pracujících se širokým kontextem přinesly transformery.

Na obr. 6 je obecná struktura transformeru, která však nemusí být vždy komplexní, Záleží na tom, pro jaký druh úlohy se používá:

1. Struktura obsahující jen enkodér, která je vhodná pro klasifikaci vstupních symbolů do jedné ze tříd, jako je např. úloha automatického doplňování interpunkce či úloha named entity recognition.

2. Struktura obsahující jen dekodér, která je vhodná pro generování výstupu "z ničeho"(prakticky ze šumu, popř. další dodatečné informace, do jaké třídy má vygenerovaný výstup přináležet).

3. Struktura obsahující enkodér-dekodér, která je vhodná pro zpracování vstupní posloupnosti symbolů na výstupní posloupnost symbolů. Představiteli takové úlohy jsou např. strojový překlad nebo sumarizace.



Obrázek 6: Struktura Transformeru převzato a upraveno z článku *Attention Is All You Need* [12])

Při trénování se enkodér snaží zakódovat celou vstupní posloupnost symbolů (větu zdrojového jazyka, která má být přeložena) na základě informace, kterou dostává od učitele (překlad věty v cílovém jazyce). Dekodér se pak snaží vstupní informaci, kterou dostává od učitele (dosud přeloženou část věty cílového jazyka), společně s informací od enkodéru snaží využít k predikci dalšího výstupního symbolu (dalšího slova překladu). Směr zpracování posloupnosti symbolů může být, zejména u enkodéru, jak zleva doprava tak i zprava doleva (pokud je pravý kontext k dispozici).

Klíčovou roli u transformeru hraje attention vrstva. Jak z názvu této vrstvy vyplývá, jejím cílem je si zapamatovat důležité události, které přicházejí na její vstup v podobě vektoru příznaků (angl. input embedding) $\boldsymbol{w}$ vytvářeného nižšími vrstvamih sítě, a na důležité z nich napřít větší pozornost než na ty méně nedůležité. Při příchodu do attention vrtvy sítě je pak každý vstupní vektor $\boldsymbol{w}$ zakódován do tří vektorů: do vektoru otázky $\boldsymbol{q}$ (query), do vektoru klíče

$\boldsymbol{k}$ (key) a do vektoru hodnot $\boldsymbol{v}$ (value). Předpokládejme, že všechny vektory $\boldsymbol{w}$, $\boldsymbol{q}$, $\boldsymbol{k}$ a $\boldsymbol{v}$ jsou řádkové. Pokud je na vstupu sítě N symbolů (slov) tvořících celý kontext (větu), je na vstupu attention vrstvy N vektorů $\boldsymbol{w}$, které můžeme uspořádat do matice $\boldsymbol{W}$, jejíž N řádek tvoří vektory $\boldsymbol{w}$, každý z nich odpovídající jedné řádce matice.

Pokud je dimenze vektoru $\boldsymbol{w}$ M, je úkolem trénování se naučit 3 matice $\boldsymbol{Q}_W$ (queries), $\boldsymbol{K}_W$ (keys) a $\boldsymbol{V}_W$ (values), které transformují každý vstupní vektor $\boldsymbol{w}$ na vektory $\boldsymbol{q}$, $\boldsymbol{k}$ a $\boldsymbol{v}$. Přitom počet řádek matic $\boldsymbol{Q}_W$ a $\boldsymbol{K}_W$ musí být roven dimenzi vstupního vektoru $\boldsymbol{w}$, tj. M. Počet sloupců matic $\boldsymbol{Q}_W$ a $\boldsymbol{K}_W$ musí být stejný.

Výslednou attention $A(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ pak můžeme spočítat ke všem vstupním vektorům $\boldsymbol{w}$, tj. k celé matici $\boldsymbol{W}$, dle následujícího vztahu:

$$\boldsymbol{Q} = \boldsymbol{W}\boldsymbol{Q}_W \tag{1}$$

$$\boldsymbol{K} = \boldsymbol{W}\boldsymbol{K}_W \tag{2}$$

$$\boldsymbol{V} = \boldsymbol{W}\boldsymbol{V}_W \tag{3}$$

$$A(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^{\boldsymbol{T}}}{\sqrt{d_k}})\boldsymbol{V}. \tag{4}$$

Proporcionální člen $1/\sqrt{(d_k)}$ se přidává kvůli stabilitě výpočtu, kde d_k je dimenze vektoru klíče (počet sloupců matice $\boldsymbol{K}_W$). Softmax se pro každou řádku své vstupní matice provede zvlášť. Výsledkem je matice, jejíž rozměr je shodný s maticí $\boldsymbol{V}$.

Protože stejné vstupní vektory umístěné na různých pozicích ve vstupní posloupnosti symbolů (každý symbol je reprezentovaný vektorem) je třeba od sebe odlišit, přidává se do modelu informace o pozici vstupního vektoru, a to tzv. pozičním kódováním (angl. positional encoding). (Toto neplatí v případě sítě modelu wav2vec 2.0, viz dále).

Velmi důležitou vlastností transformerů je možnost učení enkodéru bez učitele, čehož se právě využívá v případě modelování audia, konkrétně modelu wav2vec 2.0 [1]. Pro trénování sítě se pak dá použít obrovské množství dat (data se nepotřebují opatřovat informací od učitele) a předtrénovaný (pre-

trained) transformer se jen ve fázi tzv. *fine-tuningu* "dotrénuje"na malé množině označených dat.

4.4 Wav2vec 2.0 model

Wav2vec je model [1] založený na transformerech využívající tzv. embedding, tj. reprezentaci obrazu popisované jednotky (části audio signálu) vektorem (reálných čísel), který odpovídá bodu v obrazovém (latentním) prostoru malé dimenze a který dobře popisuje charakteristiku popisované jednotky. Přitom by tento vektor měl odlišovat jednotky s různou charakteristikou (např. jeden foném od druhého, neřeč od řeči apod.). Tzn., že vektory podobných jednotek mají ve vektorovém obrazovém latentním prostoru (téměř) stejný směr, vektory různých jednotek pak mezi sebou svírají větší úhel.

Vlastního embeddingu je dosahováno průchodem původního (neebeddingovaného) obrazu jednotky (s jejím kontextem) vrstvami neuronové sítě. Prakticky je pak vektor spočítán jako výstup nějaké skryté vrstvy neuronové sítě.

Embedding je možno aplikovat na různé úlohy - např. reprezentaci slov pro NLP (např. word2vec), pro TTS (text2vec pro následný vec2wav) nebo zvuku (wav2vec, wav2vec 2.0).

Embedding je ale možno provádět na různé úrovni využití kontextu: Například v úlohách slovního embeddingu NLP je možno využít word2vec, ELMo nebo BERT a jiné. Rozdíl mezi nimi spočívá v tom, že word2vec (CBOW, skip-grams) je víceméně statický – v zásadě si všímá jen blízkého kontextu (několika slov před a po daném slovu, jehož embedding se hledá, tj. všechna slova v textu se stejným pravým a levým kontextem délky několika málo slov mají stenou vektorouvou reprezetaci v latentním prostoru,, tj. mají stejný embedding. Sofistikovanější metodou slovního embeddingu je ELMo (Embeddings from Language Models), který využívá hluboké neuronové obousměrné (biderictional) LSTM (Long Short-Term Memory sítě a tedy embedding slova pak více zohledňuje dynamický kontext, tj. slova se stejným statickým kontextem (word2vec) umístěná na různých místech v textu mají obecně jinou reprezentaci v latentním prostoru, tedy mají obecně jiný emdedding, neboť se liší svým širším kontextem. Ještě mocnějším modelem je BERT (Bidirectional Encoder

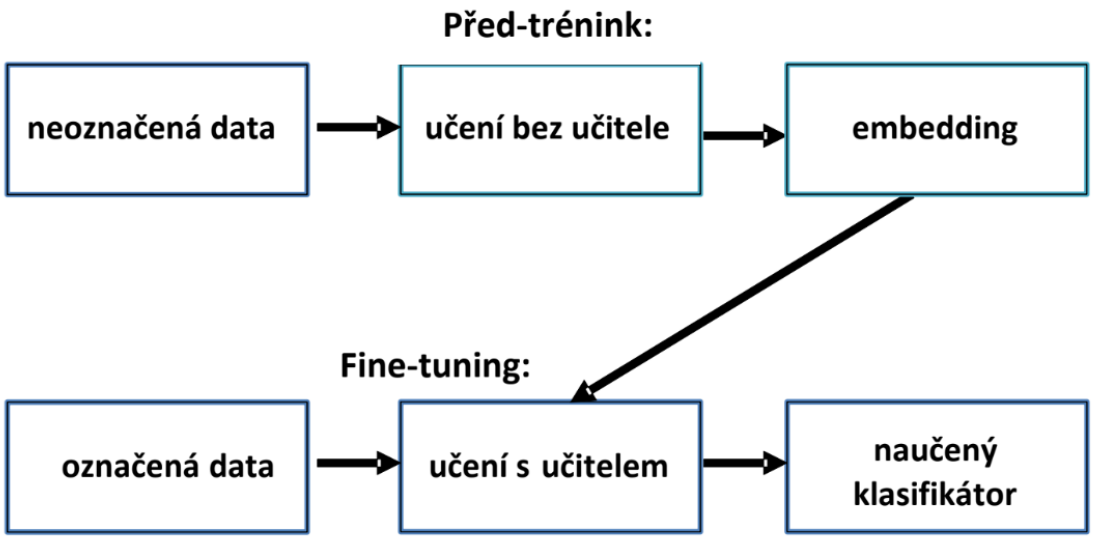
Representations from Transformers), který sice staví na myšlence obousměrného průchodu ELMo, ale pro výpočet embeddingu slov používá novější architekturu - transformery. Bylo prokázáno, že BERT produkuje vynikající slovní embedding a dosahuje state-of- the-art výsledků v různých úkolech NLP.

Wav2vec 2.0 embedding využívá právě tuto nejmodernější architekturu, podobně jako slovní embedding BERT, ale přidává k ní ještě další vylepšení.

Na proces embeddingu můžeme pohlížet jako na proces shlukování jednotek v latentním obrazovém prostoru. Protože shlukování je možno provádět bez informace od učitele, jedná se proces o učení bez učitele (unsupervised training). To je výhodné, protože pro trénink se nepotřebují označená data. Označování dat učitelem (labeling) je nejnáročnější práce při vytváření trénovacích korpusů pro strojové učení.

Máme-li k daným jednotkám jejich embedding, můžeme jejich embedding použít jako výbornou reprezentaci jednotek v další fázi vlastního trénování klasifikátoru, jehož vstupem jsou již tyto embeddingy. Takovýto klasifikátor pak pro své trénování potřebuje již jen malé množství trénovacích dat označených („olabelovaných“) učitelem. Této fázi se říká fine-tuning, během kterého se pomocí označených dat naučí model předpovídat konkrétní slova, fonémy nebo jiné třídy, do kterých chceme klasifikovat.

Díky tomuto způsobu zpracování se nepotřebuje velké množství označených dat, resp. trénování s daným disponibilním množstvím označených dat dává výsledky srovnatelné s trénováním bez použití embeddingu ale s mnohanásobným (cca sto násobným) množstvím označených dat. Fáze trénování, kdy se trénuje embedding, se nazývá před-trénink (pre-trainig) a fáze trénování klasifikátoru s učitelem, jak již bylo řečeno, se nazývá fine-tuining.



Obrázek 7: popis schéma modelu wav2vec 2.0

Autoři architektury wav2vec 2.0 předtrénovali model na obrovském souboru dat LibriVox. Poté použili celou datovou sadu Libri Speech k fine-tuningu, což vedlo k 1,8 % Word Error Rate (WER) na testovacích čistých nezašuměných datech a 3,3 % WER na testovacích zašuměných datech. Použití téměř 10krát méně dat umožnilo získat 2,0 % WER na testovacích čistých a 4,0 % na testovacích zašuměných datech. Použití pouze 10 minut označených trénovacích dat, což nejsou téměř žádná data, vedlo k 4,8 % WER na testovacích čistých a 8,2 % WER na testovacích zašuměných datech Libri Speech WSJCAM0 [1].

4.4.1 Architektura wav2vec 2.0 modelu

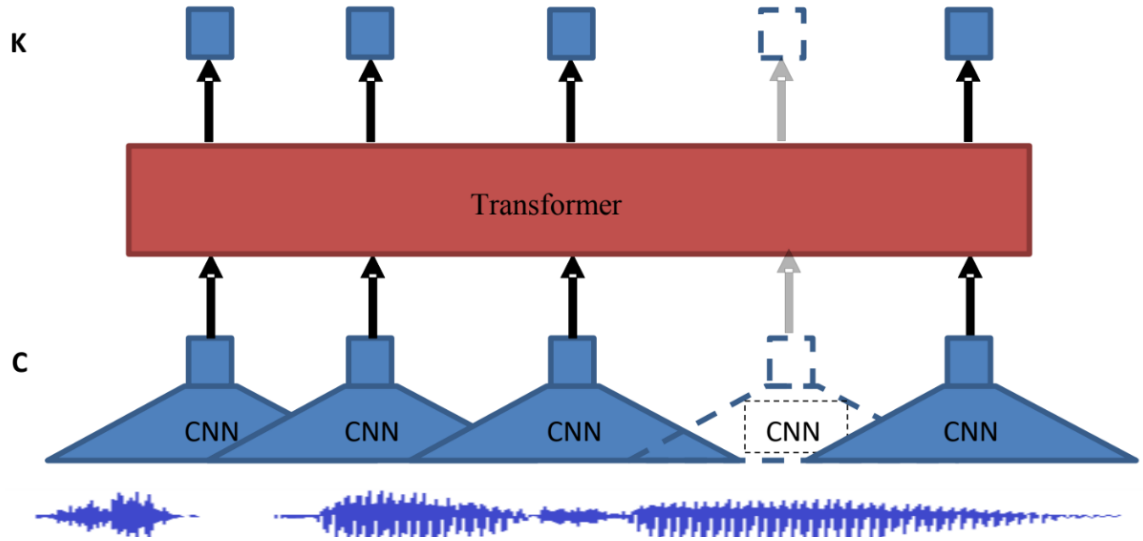
Model Wav2vec se skládá ze tří hlavních částí:

1) příznakového (feature) enkodéru, tj. několika časových konvolučních vrstev, které zpracovávají vstupní navzorkovaný audiosignál za účelem získání latentní reprezentace zvukového vstupu - C,

2) transformeru vytvářejícího reprezentaci akustické jednotky při uvažování širšího kontextu - K,

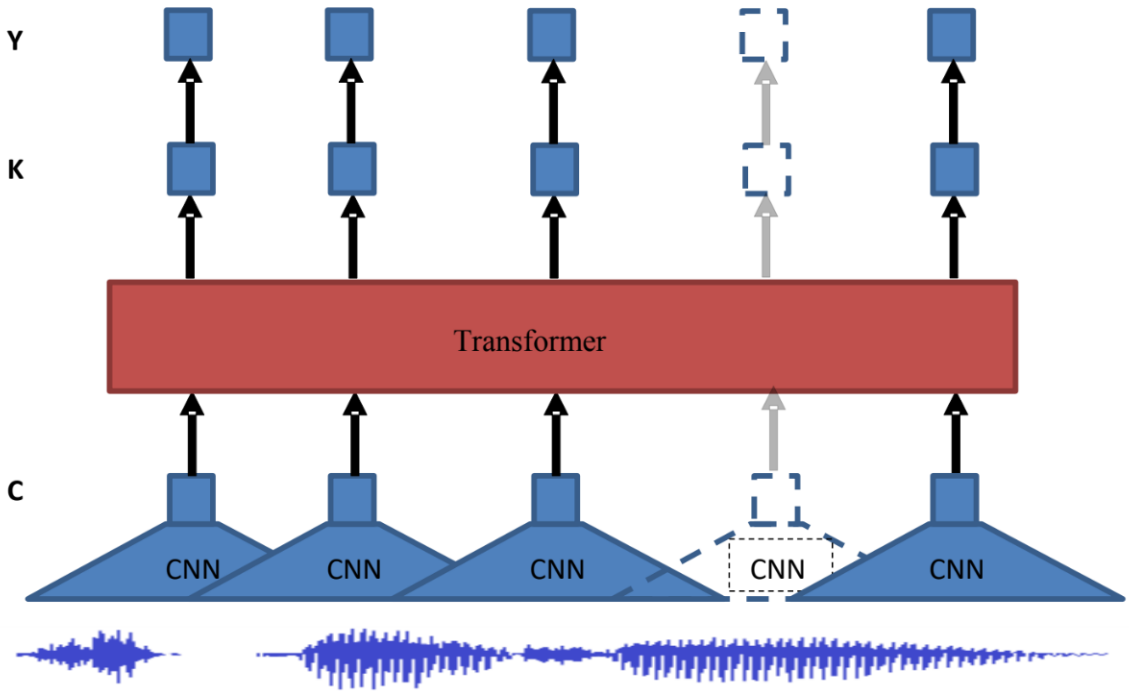
3) lineární projekce na výsledný výstup - Y.

Na obr. 8 je architektura sítě provádějící vlastní embedding z neoznačených dat.



Obrázek 8: popis schéma transformerů

Na obr. 9 je znázorněna architektura modelu užitého při inferenci, kdy je na konec sítě přidána lineární projekce Y provádějící cílovou klasifikaci.



Obrázek 9: popis schéma transformeru 2 část

4.4.2 Příznakový enkodér:

Vstupní signál je normalizován na nulovou střední hodnotu a jednotkový rozptyl. Enkodér obsahuje několik bloků časové konvoluční vrstvy následované normalizační vrstvou a GELU aktivační funkcí. Normalizace (skryté) vrstvy sítě se provádí dle vztahu:

$$y(i) = \frac{a(i) - \mu}{\delta} \tag{5}$$

$$\mu = \frac{1}{N} \sum_{i=1}^{N} a(i) \tag{6}$$

$$\delta = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (a(i) - \mu)^2 + \epsilon} \tag{7}$$

kde $a(i)$ je výstup i-tého neuronu nenormalizované vrstvy (jejíž výstup bude zpracován normalizační vrstvou) a ϵ malé číslo, např. 10^{-10} kvůli numerické stabilitě výpočtu.

GELU (Gaussian Error Linear Unit) aktivační funkce (standardizovaná

gaussovská kumulativní distribuční funkce $erf(x)$ argumentu x násobená argumentem x) má hladší průběh než aktivační funkce ReLU (Rectified Linear Unit):

$$GELU(x) = x\frac{1}{2}(1 + erf(\frac{x}{\sqrt{2}})) \tag{8}$$

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt, \tag{9}$$

kde erf značí (gaussovskou) chybovou funkci. Celkový krok (stride) enkodéru určuje počet časových kroků, jež tvoří vstupy dalšího bloku - transformeru, na obr. 8 a obr. 9 blok s názvem Transformer. Výstup enkodéru je přiváděn na vstup kontextové neuronové sítě – transformeru. Hlavním přínosem tranformeru je zachycení kontextové informace. Na rozdíl od obvyklé architektury transformeru, wav2vec 2.0 transformer používá místo pevné polohové informace (positioningu), která kóduje absolutní polohovou informaci, konvoluční vrstvu, která funguje jako relativní positioning. Výstup této konvoluční vrstvy s následnou GELU aktivační funkcí je přidán k dosavadnímu vstupu a poté je aplikována normalizace celé této vrstvy. V zásadě je pak úloha modelování rozdělena na dvě úlohy: učení lokálních vztahů v rámci malého kontextu s konvolučními vrstvami a učení globální sekvenční struktury pomocí vrstev transformeru. Toto rozdělení zjednodušuje optimalizaci transformeru, což vede ke stabilnějšímu tréninku a lepším výsledkům, protože nepotřebujeme nutit nižší vrstvy transformeru, aby se učily místní závislosti [8].

4.4.3 Princip učení ve fázi před-tréninku

Hlavní myšlenka před-tréninku je podobná přístupu uplatňovanému u modelu BERT používaného převážně pro zpracování textu (např. i pro úlohu automatické interpunkce, automatického doplnění měkkého či tvrdého i/y, určitých úloh NLP atd.): část vstupu transformeru je maskována a cílem je pak uhodnout vektorovou reprezentaci latentního příznakového (feature) vektoru k této vymaskované části. Protože uhodnout vymaskovanou část zvukového sig-

nálu zcela přesně je nemožné (jedná se o zobrazení 1:N, kde N je počet všech možných realizací vymaskované části navzorkovaného signálu, která může být považována za reálnou), jednoduchá myšlenka maskování musí být doplněna kritériem shody vymaskované části s jejím odhadem. Wav2vec 2.0 zde používá přístup kontrastivního učení. (Poznámka: původní model BERT byl též modifikován kritériem kontrastivního učení [7].)

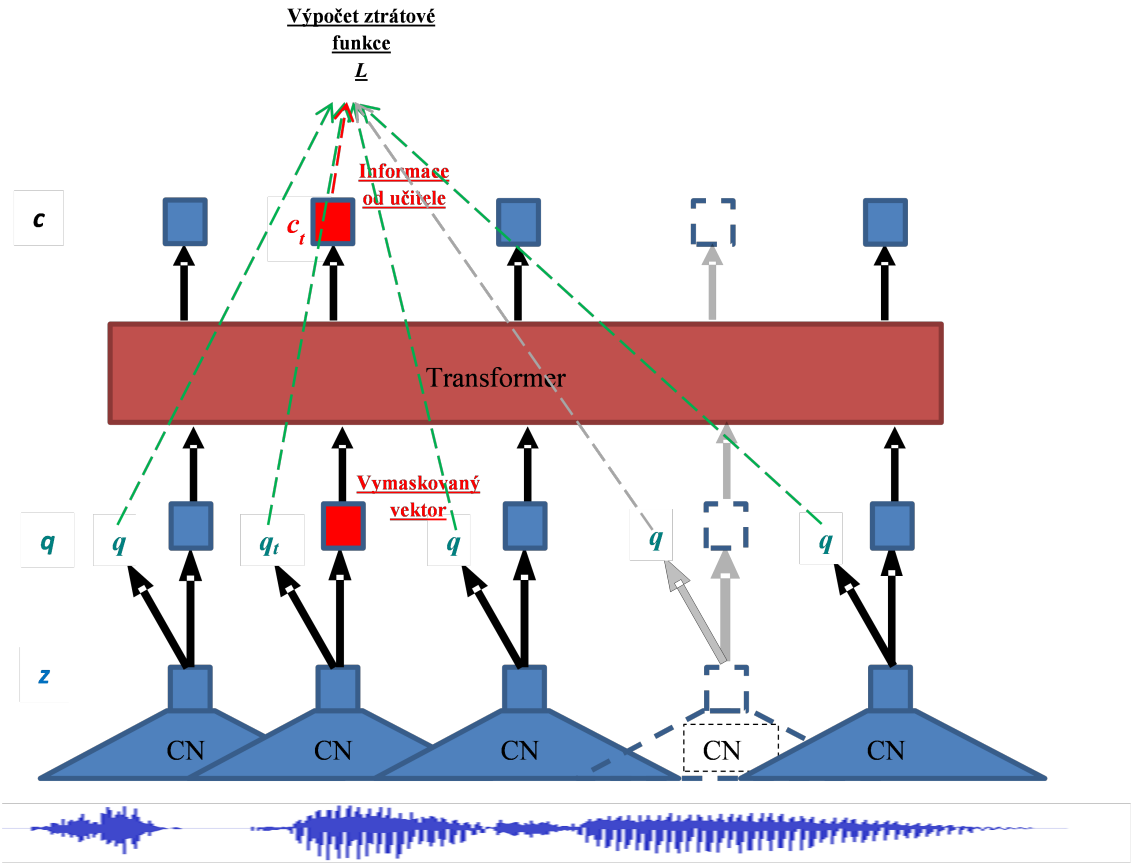
4.4.4 Kontrastivní učení (contrastive learning)

Myšlenka kontrastního učení spočívá v hledání podobných charakteristik (embeddingu, vektorů v latentním prostoru) podobných objektů (objektů stejné třídy) a s požadavkem, aby se embedding různých tříd naopak co nejvíce lišil. Princip kontrastního učení se dá velmi dobře demonstrovat na problému počítačového vidění, např. na metodě SimCLRv2, jedné z nejmodernějších kontrastivních učebních přístupů navržených týmem Google Brain, kdy se ve fázi učení neuronové síti postupně předkládají páry obrázků stejné třídy (často se používá i augmentace dat) a páry, kde každý člen páru pochází ze třídy jiné.

Podobně, jako člověk se pouhým pozorováním objektů může naučit rozpoznávat rysy (příznaky) těchto objektů a jejich rozdíly v našem světě, kontrastivní učení umožňuje modelu strojového učení zaměřit pozornost na to, které páry (latentních vektorů příznaků) jsou si podobné a které jsou odlišné, a to ještě dříve, než bude objekty klasifikovat (s použitím informace od učitele). Zde je velmi důležité si uvědomit, že kontrastivní učení spadá do oblasti učení bez učitele a tedy je možné jej aplikovat ve fázi před-tréninku.

Aby bylo možno pro trénink parametrů modelu vytvořit (diskrétní) páry latentních vektorů příznaků, uplatňuje metoda wav2vec 2.0 kvantizaci vektorů příznaků.

4.4.5 Kvantizace



Obrázek 10: schéma popisující kvantizaci vektorů příznaků

Kvantizace se provádí ve fázi pre-trainingu a to na úrovni latentních vektorů příznaků, tj. vektorů tvořících výstup enkodéru. Latentní vektory příznaků $\boldsymbol{z}$ dimenze d, vytvořené enkodérem z trénovací množiny jsou diskretizovány do položek tzv. kódových knih. Každá z G knih obsahuje V položek $\boldsymbol{w}$ dimenze d. V každé kódové knize se pak ve fázi trénování nalezne k latentnímu vektoru příznaků z spočítaného enkodérem „nejbližší“ položka $\boldsymbol{w}$ této kódové knihy. Výsledný diskretizovaný vektor $\boldsymbol{q}$ dimenze f je pak tvořen spojením všech G takto nalezených položek kódových knih do vektoru s dimenze d x G a následným uplatněním lineární transformace $\mathcal{R}^{d \text{ x } G} \rightarrow \mathcal{R}^{f}$. Výsledný počet všech kódových položek všech kódových knih je G x V (při použitých parametrech $G = 2$ a $V = 320$ tedy 102 400 diskrétních vektorů). Pro určení, k jakému

diskrétnímu vektoru $\boldsymbol{w}$ přiřadit latentní vektor příznaků $\boldsymbol{z}$ vystupující z enkodéru, je spočítán odhad pravděpodobnosti tohoto přiřazení podle Gumbelovy softmax funkce:

$$p_{g,v} = \frac{e^{(l_{g,v}+n_v)/\tau}}{\sum_{i=1}^{V} e^{(l_{g,v}+n_i)/\tau}}$$

kde $l_{g,v}$ jsou takzvané logits spočítané z výstupu enkoderu $\boldsymbol{z}$ dle kosinové podobnosti (vzdálenosti) mezi $\boldsymbol{z}$ a v-tou položkou $\boldsymbol{w}_{g,v}$ kódové knihy g:

$$l_{g,v} = \frac{\boldsymbol{z} \cdot \boldsymbol{w}_{g,v}}{\|\boldsymbol{z}\|\|\boldsymbol{w}_{g,v}\|} \tag{10}$$

$$n_i = -log(-log(u_i)), \tag{11}$$

kde u_i je realizace náhodné veličiny s rovnoměrným rozdělením $U(0,1)$ a τ je teplota (na začátku tréninku se volí např. 2 a postupně klesá na 0,5).

Gumbel-Softmax rozdělení je spojité pro $\tau > 0$, a proto má dobře definovaný gradient. Tím, že nahradíme kategorické realizace Gumbelovo-Softmaxového rozdělení, můžeme použít zpětné šíření k výpočtu gradientů. Tento postup nahrazení nediferencovatelných kategoriálních vzorků diferencovatelnou aproximací během tréninku se označuje jako Gumbel-Softmax estimátor [4]. „Straight-through“ estimátor pak znamená, že diferencovatelnou proměnnou používá pouze zpětné šíření gradientu, dopředný průchod stále používá kategorickou proměnnou.

V dopředném šíření se provede výběr dle kritéria maximální aposteriorní pravděpodobnosti, přitom apriorní pravděpodobnost má rovnoměrné rozdělení:

$$v^* = argmax_v(p_{g,v}) \tag{12}$$

Ve zpětném šíření je pak spočítán gradient Gumbelovy-Softmaxové funkce.

4.4.6 Ztrátová funkce

Při trénování sítě se nakonec snažíme maximalizovat podobnost dvou vektorových reprezentací minimalizací kontrastní ztrátové funkce L_m. Kromě ní se ještě přidává jeden vážený člen vahou α, a to diversity loss (ztrátová funkce

„diversnosti/rozmanitosti“).

$$L = L_m + \alpha L_d \tag{13}$$

$$L_m = -log \frac{exp(l_{\boldsymbol{c}_t,\boldsymbol{q}_t}/\tau)}{\sum_{\hat{\boldsymbol{q}} \in Q_t} exp(l_{\boldsymbol{c}_t,\hat{\boldsymbol{q}}}/\tau)}, \tag{14}$$

kde použitá hodnota teploty $\tau = 1$ a

$$L_d = \frac{1}{GV} * (-H(p_g)), \tag{15}$$

$$L_d = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} p_{g,v} log(p_{g,v}), \tag{16}$$

$$H(x) = -\sum_{x} P(x) log(P(x)). \tag{17}$$

Regulační člen L_m je podobný softmax funkci, ale jeho argumentem je míra $l_{\boldsymbol{c}_t,\boldsymbol{q}_t}$, která udává kosínovou podobnost vektorů $\boldsymbol{c}_t$ a $\boldsymbol{q}_t$ kde $\boldsymbol{q}_t$ je vektor skutečné kvantované latentní reprezentace řeči z množiny $K + 1$ kvantovaných reprezentací $\boldsymbol{q} \in Q_t$, která zahrnuje $\boldsymbol{q}_t$ a K distraktorů, tj. vektorů nesprávných reprezentací, a $\boldsymbol{c}_t$ je odhad vektoru $\boldsymbol{q}_t$, tj. odhad, který model provádí na základě kontextové sítě. L_d je regularizační člen nutící model maximalizovat entropii jevu vybrání nějakého vektoru ze všech kódových knih. Tento člen tedy povzbuzuje model, aby využil všech kódových slov (maximální entropii mají jevy se stejnou pravděpodobností). Maximalizace se rovná minimalizaci záporu entropie, což je ztráta diverzity.

4.4.7 Fine-tuning

Během této fáze trénování jsou konvoluční vrstvy většinou neměněny (zamraženy, freezed). Není užita žádná kvantizace, místo ní je přidána lineární projekce na konec sítě (kontextová reprezentace C vstupu – viz obr. 9). Model je pak „fine-tunován“ učením s učitelem (např. každý úsek audiosignálu má přiřazen značku řeč / neřeč).

4.4.8 Wav2vec 2.0 modely užité v této práci

Základním modelem wav2vec 2.0 užitým v této práci byl model s architekturou BASE uvedenou v článku *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations* [1]. Kromě modelu s architekturou BASE jsem použil ještě model s podobnou architekturou, ale s menším počtem parametrů v podobném poměru jako je mezi modely LARGE a BASE uvedené v původním článku. Menší model bude značen v dalším textu jako SMALL. Oba modely byly připraveny Ing. Janem Lehečkou, Ph.D. za použití platformy Huggingface . Konfigurační soubory ve formátu podporovaného touto platformou popisující architektury sítí obou dvou modelů jsou uvedeny v Příloze C. Zde pouze uvedu hlavní parametry architektury BASE. Vstupem je patnáctivteřinový audiosignál vzorkovaný na 16 kHz. Příznakové vektory, do kterých se vstupní data kódují, jsou vytvářeny enkodérem, který je tvořen sedmi bloky s časovými konvolučními vrstvami, které mají 512 kanálů s krokem (stridem) (5,2,2,2,2,2,2) a rozměrem kernelu (10,3,3,3,3,2,2). Výsledkem je 749 floatových logits odpovídajících frekvencí 50 Hz s krokem (stridem) 20 ms mezi vzorky. Receptivní pole má cca 25 ms (cca 400 vzorků). Konvoluční vrstva nahrazující standardní *positioning* transformeru má velikost jádra 128 a 16 skupin (počet kernelů v jedné vrstvě). Model obsahuje 12 transformerových bloků s dimenzí (hidden size) 768, vnitřní dimenze (intermediate size) 3 072 a 8 attention heads. Velikost dávky (batche) je při trénování třicet vteřin, při inferenci patnácti vteřin [1]. Experimentálními testy provedenými ing. Janem Lehečkou, Ph.D. a ing. Alešem Pražákem,Ph.D. bylo zjištěno, že obě sítě dosahují nejpřesnějšího výsledku na prostřední třetině vstupního signálu, tj. prostředních pět vteřinách. Proto ve všech dále uvedených experimentech, jejich výsledcích i implementacích se na vstup každé sítě vždy přivádí 15 vteřin audiosignálu. Síť pak k patnácti vstupním vteřinám vrací svůj výstup ve formě 749 (platí pro BASE) nebo 299 (platí pro SMALL) logits, ale s nich se dále počítá jen s jejich prostřední třetinou. Audiosignál se tedy zpracovává po pěti vteřinách (časová okna zpracování audiosignálu se překrývají o 10 vteřin a posunují se o 5 vteřin zleva doprava).

5 Návrh postupu řešení

Na počátku této práce se nabízelo řešení postavené na systému NINA (viz část 4.2 NINA), popřípadě možnost přetrénování či modifikace této neuronové sítě. (Zde je třeba poznamenat, že vzhledem k tomu, že detektor NINA byl natrénován na 700 000 anotovaných hodinách trénovacích dat, asi nelze očekávat zlepšení s trénovací sadou dosahující zlomku procenta velikosti sady původní, byť z cílové domény českého jazyka). Během práce na implementaci detektoru NINA se objevily zprávy o nových metodách založených na wav2vec 2.0 modelech, které reportovaly vynikající úspěšnost rozpoznávání. Na základě těchto skutečností byl navržen následující postup prací: Postup při hledání nového cílového řešení:

1. Vyhodnotit úspěšnost a rychlost metod:
   a) NINA
   b) wav2vec 2.0
   c) Srovnat výsledky s referenčním stávajícím řešením ORG.
2. Navrhnout vhodné řešení.
3. Realizovat navržené řešení.
4. Otestovat navržené řešení na evaluačních datech.

Ohledně bodu 1.b. je třeba poznamenat, že wav2vec 2.0 modelů existuje více, např. autoři původního článku o *wav2vec 2.0* [1] popisují modely LARGE a BASE, na pracovišti školitele pak byly natrénována řada dalších modelů. Na základě experimentů předcházejících této práci z nich byla vybrány modely BASE a tzv. SMALL. Bližší popis všech modelů (ORG, NINA, SMALL, BASE) byl popsán v předcházející kapitole 4 této práce. V kapitole 7 pak budou uvedeny dosažené výsledky těchto detektorů na reálných datech. Nejdříve však bude popsána tvorba datových sad použitých při experimentech.

6 Tvorba datových sad

Pro provedení testů (vyhodnocení úspěšnosti a rychlost metod) je třeba připravit datové sady z cílové domény, tj. z audiodat vysílaných pořadů Českého rozhlasu. K dispozici byla datová sada 120 hodin audia rozhlasového vysílání, která se však použila na trénování (přesněji dotrénování = ang. fine-tuning) wav2vec 2.0 modelů. Tato sada je v této práci značena TRAIN_DATA (trénovací data). Tuto sadu nelze použít pro vyhodnocení a porovnání wav2vec 2.0 modelů s jinými modely. Pro vyhodnocení úspěšnosti různých metod bylo proto nutno připravit novou datovou sadu EVAL_DATA (evaluační data). 2.0 Zároveň je potřebné připravit i sadu pro nastavení hyperparametrů (viz kapitola: 9) cílového řešení. Tato datová sada bude v dalším textu značena DEVELOPMENT_DATA (vývojová sada). Tvorba datových sad probíhala následujícím způsobem:

1. Pro přípravu nových datových sad byla nejprve provedena rešerše dostupných anotací: Bylo zjištěno, že sice k dispozici nějaká data jsou (z období vývoje a trénování stávajícího řešení ORG), ale pro jejich využití bude potřeba jejich kontrola a úprava. Ve skutečnosti se jednalo o nejednoduchou úlohu – anotovaná data byla často bez korespondujících audiosouborů (wavů) nebo naopak existovala audiodata bez anotací, anotace byly různé kvality, různí anotátoři značili data různě a data se často řídila různými anotačními schématy).

2. Následovala revize stávajících dat, jejich očištění od chyb, zjištění anotačních schémat.

3. Pro vytvoření evaluační sady bylo nakonec rozhodnuto připravit data nová. Pro přípravu nových datových sad bylo původně uvažováno, že budou vytvořeny týmem několika anotátorů. Při jejich shánění a práci se zde, bohužel, projevil nedostatek jejich rychlosti a přesnosti. Nakonec jsem největší práci při přípravě datových sad vynaložil sám – ať již přímou anotací či kontrolou a často předěláním práce jiných. To bylo nutné z důvodu, že navzdory předpokladu, že popis, jak se mají anotace provádět, je jasný, tomu tak v praxi nebylo. V průběhu prací se vyskytla

řada případů, kdy nebylo zcela zřejmé, jak přesně daný úsek zvukového signálu označit – jedním z mnoha příkladů je hlasové povzbuzování či skandování při sportovních či jiných soutěžích, kde v některých případech lze takovýto projev považovat za řeč, jindy tomu tak již ale být nemusí.

4. Všechny datové sady musí být zkontrolovány, sjednoceny (sjednocením není myšleno, že sady po sjednocení budou mít stejná anotační schémata, ale bude konzistentně chápáno a anotováno, co je a co není řečový úsek a případně upraveno).

5. Výše uvedená práce byla velice časově náročná. Někteří anotátoři nebyli schopni vůbec začít práci, jiní vyžadovali velkou podporu a pracovali velmi pomalu a nepřesně. Příprava a kontrola nových datových sad proto spočívala především na práci autora této diplomové práce.

6.1 Typy datasetů

V této práci byly postupně užity soubory datasetů rozdělené do tří skupin: Trénovací dataset, vytvořený před začátkem této práce, nastavující a evaluační dataset vytvořené autorem této práce.

6.1.1 Trénovací dataset TRAIN_DATA

První dataset čítající 90 souborů se 120 hodinami anotovaných audionahrávek převážně ze stanic ČRo Dvojka, Plus a Radiožurnálu byl použit již pro natrénovaní modelů BASE a SMALL ing. Janem Lehečkou, Ph.D. Různorodost tohoto datasetu má zaručovat dostatečnou robustnost modelů. Nejdříve však bylo potřeba dataset analyzovat a zjistit, jakého formátu jsou data, jak byly označeny úseky řeči a jak byly označeny úseky neřečových částí. Výstupem anotací jsou *trs* soubory ve formátu Transcriber[2] popisující jednotlivé úseky audiostop. Množina značek (tagů), kterou jsem následně nechal považovat za indentifikátor řeči, v tomto datasetu byla následující: (*P, SPEECHBACKG, SPEECH, NEstudio, podkres, studio, Slovensky, telefon, reklama, en, ru, sk*). Data jsem ale ještě kontroloval, např. všechny reklamy jsem dělil na místa s výskytem řečí a bez řeči.

Označování (tagování) úseků bez řeči bylo následující: (*S, B, MUSIC, MUSICSONG, předěl, hudba*). Problém nastával, bylo-li anotátorem přiřazeno více značek (tagů) k jednomu zvukovému úseku. V tom případě se musel vyhodnocující program analyzující datasety rozšířit o možnost prioritních tagů. Bylo implementováno pravidlo, že vyskytuje-li se k danému úseku řečový i neřečový tag, je tento daný úsek dále uvažován jako řeč na neřečovém podkresu a úsek je klasifikován jako řečový signál.

6.1.2 Nastavující dataset DEVELOPMENT_DATA

Tento dataset čítá 100 souborů, kde každý soubor obsahuje přibližně jednu hodinu vysílání s malým přesahem před začátkem a po konci hodiny (cca 15 - 20[s]). Tento dataset se zaměřil především na stanici Radiožurnál, Dvojka a Plus, ale obsahuje i pořady ze stanice Vltava, Radio Wave, Regína a Brno. Anotace byla vytvořena autorem této práce a pravidla a postupy anotací, stejně

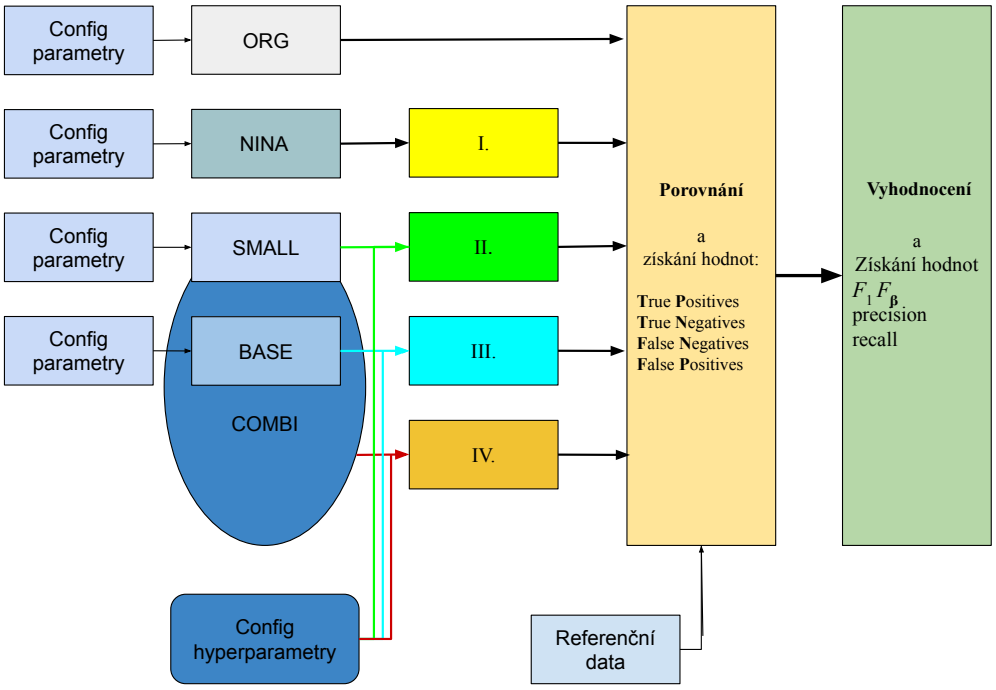
tak jako popisy problémových částí vysílání jsou k nahlédnutí v příloze této práce. Aby šla anotační práce rychleji, použil jsem na předznačení audioúseků pomalý BASE detektor a automatické anotace pak ručně opravoval. Malá rychlost BASE modelu nevadila, člověk pracoval pomaleji (na rozdíl od stroje musí odpočívat) a tak nemusel na předzpracování dat čekat. Vzhledem k tomu, že byla užita část předzpracování počítačem, bylo již vytvoření řečových a neřečových úseků jednotnější než v případě trénovací datové sady TRAIN_DATA (má podobné začátky a konce u podobných znělek a přechodů z řeči na "neřeč" a obráceně). Byly vybírány různorodé časy vysílání a zejména pak ohlídáno, aby se tatáž relace daného pořadu nevyskytovala ve vytvářeném datasetu vícekrát a nedošlo tím k rozvážení této potenciálně trénovací množiny, kterou se do budoucna i tento dataset může stát. Datová sada je k dispozici pro v výzkumné účely na úložišti Katedry kybernetiky ZČU v Plzni.

6.1.3 Evaluační dataset EVALUATE_DATA

Tento dataset je nejmenší, obsahuje 20 anotovaných hodin ze všech dostupných stanic Českého rozhlasu z různých časů vysílání a podobně jako soubory v nastavujícím datasetu obsahují jeho soubory pouze značky (tagy) S ("neřeč") a P (řeč) určující, o jaký úsek se jedná. I tento dataset byl předzpracován detektorem BASE, aby byla ucelena jednota počátků. Vzhledem k tomu, že se jedná o poměrově nejrůznorodější dataset, bylo jeho anotování nejsložitější a nejnáročnější a zabralo poměrově nejdéle času.

6.2 Druhy vyhodnocovacích skriptů

Zpracování a vyhodnocení bylo prováděno skripty v programovacím jazyce Python. Každý detektor (ORG, NINA, SMALL a BASE) dovede zpracovat audiosoubor každého výše uvedeného datasetu. Výstup z každého detektoru je jiný. Proto byly nejprve vytvořeny skripty, které výstup každého detektoru převedou na jednotný formát, za který byl zvolen formát Transcriberu s příponou *trs*. Poté se na data aplikuje program, který vzniklou dávku *trs* souborů porovná s referenční dávkou odpovídajících *trs* souborů. Porovnávací program tedy porovnává mezi sebou vždy 2 *trs* soubory (i poměrně různorodého, tj, ne zcela standardního formátu *trs*, ve kterém byla zapsána řada anotací nejspíše proprietárními programy pracoviště sloužícími k přípravě trénovacích dat TRAIN_DATA). Testování jednotlivých detektorů neznamenalo jen jejich samostatné testování, ale i testování jejich kombinací, jak je ukázáno v kapitole 9, což vede z důvodu existence velkého množství konfigurací na rozsáhlé výpočty. Z těchto důvodů byly programy rozděleny na dvě skupiny (etapy), první z může počítat zcela samostatně (např. výpočet výstupních pravděpodobností či míry důvěry ve svůj výstup) a druhá z nich již vyžaduje data zpracovaná programy z první části pro výpočet výsledku závislého na konfigurací více detektorů. Tímto postupem (princip dynamického programování) se dosáhlo snížení počtu komplexních výpočtů. Na obr. 11 je pak grafické znázornění procesu pro vyhodnocování datasetů. V popisu uvedu jen hlavní vytvořené skripty obou skupin.



Obrázek 11: popis schéma workflow skriptů

Legenda obr. 11:

I. NINA2TRS.py viz sekce 6.2.1

II. "*_COMBI_VAD_ProbProb2trs.py*"viz sekce 6.2.3

III. "*_COMBI_VAD_ProbProb2trs.py*"viz sekce 6.2.3

IV. "*_COMBI_VAD_ProbProb2trs.py*"viz sekce 6.2.3

Config. parametry obsahují cesty vstupních souborů

Config hyperparametry obsahují různé nastavení vah, dva pro SMALL model a jednu váhu pro BASE model

Referenční data obsahují anotované datasety viz. kapitola 6.1

Combi značí skript *COMBI_Compute_prob.py* viz sekce 6.2.2

6.2.1 NINA2TRS.py

Prvním skriptem je skript pro převod výstupu programu NINA do souboru formátu *trs*. Tento skript konvertuje výstup z inaSpeechSegmeter na formát *trs*, přičemž zachovává všechny tagy vstupního souboru v souboru výstupním (tj. nedochází k jejich přeznačení na jiné).

6.2.2 COMBI_Compute_prob.py

Tento skript slouží pro přípravu dat pro vyhodnocení detektoru založeného na wav2vec 2.0 modelech. Skript využívá vstupní textový soubor pro určení cesty a užití BASE nebo SMALL modelu. Načítají se postupně wav soubory, ke kterým se vypočítává časová posloupnost pravděpodobností jevu, že v daném čase vstupní wav soubor obsahuje řeč. Výstupní pravděpodobnost se vypisuje (ukládá) s různou frekvencí, např. model BASE vyprodukuje jednu hodnotu (pravděpodobnost "neřeči", tj. nízká hodnota znamená řeč, hodnoty blížící se k 1 značí řeč) ke každé 50-tině vteřiny vstupního wav souboru, model SMALL produkuje jednu hodnotu jen každou 20-tinu vteřiny. Skript využívá knihovny pytorch, torchaudio, tranfromers knihovny a backendu soundfile. Výstupní pravděpodobnosti se ukládají do stejného adresáře, ze kterého jsou načítané vstupní soubory wav (výstupní soubory mají příponou _prob.txt).

6.2.3 _REF_COMBI_VAD_fromProbProb_comparison.py

Provádí v zásadě kód dvou programů.

První z nich je *_COMBI_VAD_ProbProb2trs.py* (říkejme mu A) načítá pravděpodobnosti uložené ve dvou souborech spočítaných programem *COMBI_Compute_prob.py* (viz výše). Jeden z těchto dvou souborů jsou pravděpodobnosti počítané modelem SMALL a druhý soubor obsahuje pravděpodobnosti spočítané modelem BASE). Dále program A načítá konfigurační soubor C, který obsahuje na každé své řádce hodnoty 3 hyperparametrů (tj. prahy $Th1$, $Th2$, Thb, jejichž význam je blíže vysvětlen v kapitole 9). Pro každou trojici hyperparametrů (konfiguraci hyperparametrů K) načtené z dané řádky konfiguračního souboru program A spočítá z pravděpodobností modelu SMALL a z pravděpodobností modelu BASE výstupní *trs* soubor a uloží jej do dočasného adresáře.

Druhý program : *REF_COMBI_VAD_BASEfromProb_comparison.py* (dále nazývaný B) pak porovná programem A vyrobený *trs* soubor uložený v dočasném adresáři s referenčním (ručně anotovaným) *trs* souborem. Program B zapisuje výsledek do 2 souborů: *out_Th1_Th2_Thb.txt* a *output_sum.txt*. Soubor *out_Th1_Th2_Thb.txt* program nazve tak, aby jeho jméno jednoznačně identifikovalo aktuální řádku načtené konfigurace K, tj. místo symbolů $Th1$ $Th2$ Thb dá so jména souboru kpnkrétní hodnoty hyperparametrů. Program B do souboru *out_Th1_Th2_Thb.txt* zapíše výsledek v podobě záznamu: časová délky zpracovávaného souboru, časová délky referenčního souboru, počet TP (True Positives), FN (False Negatives), TN (True Negatived), FP (False Positives) (celkový počet událostí = $TP + FN + TN + FP$ = počet zpracovaných vteřin souboru = časová délka souboru v vteřinách), doba výpočtu v vteřinách a další údaje). To udělá pro každý zpracovávaný soubor, ale zatím stále s jednou a tou samou konfigurací prahů K.

Jakmile již neexistuje žádný soubor, který by nebyl zpracován s konfigurací hyperparametrů K, tak program B sečte všechny hodnoty řádek souboru *out_Th1_Th2_Thb.txt* s konfigurací K (pro každou položku ale zvlášť!) a tento součet zapíše do jedné nové řádky souboru ouput_sum.txt. Poté načte další konfigurační řádku souboru C a postup opakuje, dokud nevyčerpá

všechny řádky souboru C.

Samostatný program B se dá použít k porovnání jakýchkoliv dvojic *trs* souborů, může tedy porovnat referenční *trs* soubor s jakýmkoliv trsem vyprodukovaným některým detektorem. Konfigurační soubor C má význam pro dávkové zpracování mnoha nastavení konfigurací prahů (řádově stovky až tisíce) a jeho využití je hlavně pro nalezení hyperparametrů modelu COMBI popsaném v v kapitole 9 této práce.

Pro případ potřeby detailnějšího popisu dodávám, že skript *_REF_COMBI_VAD_fromProbProb_comparison.py* vyžaduje temporary adresář, ve kterém se ukládají dočasné soubory **_SDC.trs* pro každé nastavení prahů definované konfiguračním souborem. Obsah temporary adresáře se přepíše vždy novými dočasnými soubory. Program načítá kromě konfiguračního souboru a kromě 2 souborů definujících řečové a neřečové tagy *Ppattern.txt* (definuje tagy řeči) a *Spattern.txt* (definuje tagy "neřeči") ještě 2 vstupní soubory definující množinu zpracovávaných souborů:
input_compute_trs_from_prob_and_prob_file_list.txt a
input_list_of_trss.txt. Je třeba dbát na to, aby poslední (3.) sloupec souboru *input_compute_trs_from_prob_and_prob_file_list.txt* byl shodný s posledním (2.) sloupcem souboru *input_list_of_trss.txt* Výstupem je množina souborů *out_Th1_Th2_Thb.txt*, každý soubor pro jednu konfiguraci hyperparametrů (prahů $Th1$ $Th2$ Thb) obsahující na každé své řádce výsledky pro jeden zpracovaný vstupní soubor s touto konfigurací. Výstupem je soubor *ouput_sum.txt*, který obsahuje všechny poslední řádky výše zmíněných výstupních souborů. Výstupem je také soubor *log.txt*, který pak obsahuje varovná hlášení.

Program B provede porovnání referenčního **.trs* souboru s **.trs* souborem spočítaným algoritmem A využívajícím algoritmus kombinace SMALL a BASE modelu. Program načítá seznam souborů určených ke zpracování z textového souboru s názvem *input_list_of_trss.txt*, který na každé své řádce obsahuje po řadě dvojici (oddělenou bílým znakem - whitespacem): *pathname* referenčního **.trs* souboru a *pathname *.trs* souboru spočítaného programem A.

V konfiguračním souboru souboru jsou uloženy po jednotlivých řádkách

float číselné hodnoty:
(i) tolerance rozdílu (rozdílu mezi zpracovávaným a refernečním) časových délek obou vstupních souborů. Zjistilo se, že soubory **.trs* se mohou z různých zdrojů zpracování svou délkou nepatrně lišit (údaje o velkosti tolerance se uvádí ve vteřinách - např. 0,1),
(ii) tolerance hranic segmentů - malá odchylka odlišnosti hranic vyhodnocovacího **.trs* souboru od referenčního **.trs* souboru by měla být tolerována, tj nevyhodnocována jako chyba,
(iii) údaj, zda se mají při vyhodnocování spojovat sousední segmenty všechny shodně označené stejným tagem P nebo S (nebo i jiným tagem, jsou-li takové) - je to z důvodu, aby se netolerovaly malé rozdíly dle bodu (ii) mezi segmenty označenými stejným tagem - hodnota 1 značí spojovat, hodnota 0 nespojovat. Program využívá pro porovnání oubou souborů soubory *Ppattern.txt* (definuje tagy řeči) a *Spattern* (definuje tagy "neřeči"). Tj. všechny tagy uvedené v *Ppattern.txt* jsou při vyhodnocování uvažovány jako řeč (P), všechny tagy v *Spattern.txt* jsou brány jako "neřeč" (S). Různé značení řeči či "neřeči" vzniká z důvodu používání rozdílných tagů z různých zdrojů anotací.

Pokud ve vstupním **.trs* souboru je k danému segmentu přiřazeno více tagů (jsou povoleny nanejvýše 3 různé tagy popisující jeden konkrétní segment vstupního audia), z nichž alespoň jeden je uveden v souboru *Ppattern.txt*, je segment interpretován jako řeč. Pokud ani jeden z těchto (maximálně) 3 tagů není uveden v *Ppattern.txt* souboru, ale alespoň jeden z nich je uveden v souboru *Spattern.txt*, je segment interpretován jako "neřeč". Pokud ani jeden z (maximálně) 3 tagů přiřazených k segmentu vstupního **.trs* souboru není uveden ani v jednom z pattern souborů (tj. v souboru *Ppattern.txt* či *Spattern.txt*), příslušný segment se nevyhodnotí (označí se ve výstupním souboru jako Others). Výstupní soubor *compare_2_trs_output.txt* obsahuje počty vteřin TP (True Positive), FN (False Negative) chyb, FP (False Positive) chyb, TN (True Negative), počty vteřin Others, počty vteřin tolerovaných hraničních úseků, délky wav ve vteřinách, použité hodnoty prahů $Th1$, $Th2$, počet volání BASE modelu (jedno volání odpovídá zpracování 5 vteřin, takže pro porovnání s počty ostatních hodnot v vteřinách je toto číslo třeba násobit pěti).

Soubor *log.txt* pak obsahuje varovná hlášení.

6.2.4 _Make_01_from_Trs.py

Program slouží k předpřípravě dat k fine-tuningu modelů SMALL a BASE. Pro každý segment audia (20-tinu vteřiny pro fine-tuning modelu SMALL, 50-tinu vteřiny pro fine-tuning modelu BASE) zapíše do výstupního souboru 0 ("neřeč") nebo 1 (řeč). Program na svém výstupu tedy nakonec vytvoří soubor znaků 0 a 1 oddělených mezerou, kde každá 0 či 1 na i-té pozici odpovídá i-tému časovému framu (délky buďto jedné 20-tině pro SMALL model nebo 50-tině vteřiny pro BASE model) audiosouboru, jehož *trs* soubor program *_Make_01_from_Trs.py* zpracovává. Tj. pokud chceme k danému wav souboru přiřadit značky (labely 0 a 1 nesoucí údaje o přítomnosti řeči či "neřeči" v daném úseku audiosouboru, musíme mít k dipozici přesně anotovaný *trs* soubor (pro fine-tuning je přesná ruční anotace vstupních dat nutná), který je vstupním souborem programu *_Make_01_from_Trs.py.* . O tom, zda se na výstup zapíše jedna 0 či 1, rozhoduje údaj vstupního **.trs* souboru (pokud se jedná o časový frame ležící v segmentu řeč, program zapisuje 1, pokud časový frame leží v segmentu "neřeč", zapisuje 0. Seznam značek (tagů) označujících řeč a seznam značek (tagy) neřečových jevů jsou zapsány v konfiguračních souborech *Ppattern.txt* s *Spattern.txt*, které se mohou dle potřeby měnit. Program je robustní i na případy, kdy není segment označen jako řeč ani jako "neřeč". V takovém případě zapíše na výstup hodnotu 0,5. Toto rozšíření bylo potřeba pro možné využití starších sad anotací, ke kterým chybí dokumentace. Pokud vstupní soubor obsahuje čísla 0,5, je nutné se pokusit provést přiřazení těchto hodnot k (jejich přepsání na hodnotu) 0 nebo 1. Pokud se hodnoty 0,5 v souboru ponechají, mohou se všechny (zpravidla třiceti vteřinové) úseky odpovídající hodnotě 0,5 v dalším kroku přípravy trénovacích dat (rozdělení na časové 30-ti vteřinové úseky pomocí programu viz dále program *_REZACKATXT.py*) vyloučit z trénovací množiny.

6.2.5 _REZACKATXT.py

Tento skript připravuje množinu trénovacích dat pro fine-tuning modelu *SMALL.txt* nebo *BASE.txt*. Jeho vstupem jsou soubory vytvoření programem

_Make_01_from_Trs.py, výstupem textové soubory obsahující posloupnost znaků složenou z 0 a 1. Délka souborů odpovídá 30 vteřinám. Jinou délku mají tedy výstupní soubory pro fine-tuning modelu BASE (749 hodnot) a jinou délku pro fine-tuning modelu SMALL (299 hodnot).

6.2.6 _REZACKA.py

Tento skript připravuje množinu trénovacích dat pro fine-tuning modelu SMALL nebo BASE. Vstupem tohoto programu je *wav* a výstupem jsou *wav* soubory o délce 30-ti vteřin.

7 Vyhodnocení experimentů

Na vytvořené sadě EVAL_DATA byla vyhodnocena výpočetní náročnost a úspěšnost každé metody, tj. ORG, NINA, BASE, SMALL. Protože dle požadavku Českého rozhlasu je při detekci větší chybou nezachytit řeč než chybně tvrdit, že daný audioúsek je řečový, přestože ve skutečnosti řeč neobsahuje, byla kromě obvyklé (symetrické) míry chyby detekce F_1 počítána i chyba F_β:

$$F_1 = 2\frac{precision \cdot recall}{precision + recall}$$

$$F_\beta = (1+\beta^2)\frac{precision \cdot recall}{\beta^2 precision + recall} \text{ , kde}$$

$$presicion = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN},$$

kde TP je hodnota True Positives, tj. počet případů správně klasifikovaných řečových úseků,

TN je hodnota True Negatives, tj. počet případů správně klasifikovaných neřečových úseků,

FP je hodnota False Positives, tj. počet případů nesprávně klasifikovaných neřečových úseků jako řečových,

FN je hodnota False Negatives, tj. počet případů nesprávně klasifikovaných řečových úseků jako neřečových.

Přitom každý případ se vztahuje k jedné vteřině záznamu datové sady (každý případ samozřejmě k jiné). Celkový počet případů (provedených klasifikací) je tedy při testu roven celkovému počtu vteřin příslušné datové sady. Hodnota *beta* byla zvolena $\beta = 2$, tj. recall je považován za dvakrát důležitější než precision.

Vyhodnocení všech metod (ORG, NINA, BASE, SMALL) bylo provedeno na EVAL_DATA datové sadě čítající 20 hodin referenčních ručně pečlivě anotovaných dat.

Výsledky výpočetní náročnosti: Doba výpočtu přepočtená na 1 hodinu

Detektor:	výpočetní doba [s]:
ORG	375,6
NINA	108,8
SMALL	136,5
BASE	3141,0

Tabulka 1: Tabulka porovnávající časy detektorů

zpracovávaného audiosignálu na i7 4790 3,6 GHz, (CPUMARK = 7239, využity 4 fyzická vlákna) je uvedena v tabulce 1. Na základě těchto výsledků se jeví jako perspektivní modely SMALL a hlavně NINA. Model BASE nesplňuje podmínky výpočetní náročnosti, neboť je více jak 8-krát časově náročnější než původní ORG detektor (prakticky použití BASE modelu znamená, že detekce řeč/neřeč jednoho rozhlasového kanálu představuje 80 % výkonu CPU i7 se čtyřmi fyzickými jádry). Provedené testy porovnávající úspěšnost detekce mezi detektory byly následující:

1. Na sadě EVAL_DATA byla spočítána F_1 míra pro všechny detektory a byly provedeny testy:
   (a) Kontrolní (pro srovnání s testem Wilcoxonovým, viz dále) levostranný, (H_0: detektor vlevo není horší (má vyšší nebo stejnou hodnotu F_1) než detektor vpravo) párový t-test.
   (b) Pro případ, že by data neměla normální rozdělení, byl spočítán Wilcoxonův levostranný neparametrický sign rank test (H_0: detektor vlevo není horší než detektor vpravo).
2. Na sadě EVAL_DATA byla spočítána F_β míra s hodnotou $\beta = 2$ pro všechny detektory a provedeny testy s podobným výsledkem jako pro hodnotu míry F_1:
   (a) Kontrolní (pro srovnání s testem Wilcoxonovým, viz dále) levostranný, (H_0: detektor vlevo není horší (má vyšší nebo stejnou hodnotu F_β) než detektor vpravo) párový t-test.
   (b) Pro případ, že by data neměla normální rozdělení, byl spočítán Wilcoxonův levostranný neparametrický sign rank test (H_0: detektor vlevo není horší než detektor vpravo).

Detektor	detektor	p-hodnota	p-hodnota po Bonferroniho korekci:
ORG	NINA	0,000 000 485	$p < 0{,}01$
ORG	SMALL (práh 0,5)	0,000 000 075	$p < 0{,}01$
ORG	BASE (práh 0,5)	0,000 000 086	$p < 0{,}01$
NINA	SMALL (práh 0,5)	0,000 000 239	$p < 0{,}01$
NINA	BASE (práh 0,5)	0,000 000 811	$p < 0{,}01$
SMALL	BASE (práh 0,5)	0,000 864	$p < 0{,}01$
ORG	NINA	0,000 000 485	$p < 0{,}01$
ORG	SMALL (práh 0,35)	0,000 000 088	$p < 0{,}01$
ORG	BASE (práh 0,35)	0,000 000 090	$p < 0{,}01$
NINA	SMALL (práh 0,35)	0,000 000 794	$p < 0{,}01$
NINA	BASE (práh 0,35)	0,000 001 03	$p < 0{,}01$
SMALL	BASE (práh 0,35)	0,000 410	$p < 0{,}01$

Tabulka 2: Tabulka výsledků levostranného párového t-testu F_1

Detektor	detektor	p-hodnota	effect size	p-hodnota po Bonferroniho korekci
ORG	NINA	0,000 000 954	0,872	$p < 0{,}01$
ORG	SMALL(práh 0,5)	0,000 000 954	0,872	$p < 0{,}01$
ORG	BASE (práh 0,5)	0,000 000 954	0,872	$p < 0{,}01$
NINA	SMALL (práh 0,5)	0,000 000 954	0,872	$p < 0{,}01$
NINA	BASE (práh 0,5)	0,000 000 954	0,872	$p < 0{,}01$
SMALL	BASE (práh 0,5)	0,000 000 954	0,872	$p < 0{,}01$
ORG	NINA	0,000 000 954	0,872	$p < 0{,}01$
ORG	SMALL(práh 0,35)	0,000 000 954	0,872	$p < 0{,}01$
ORG	BASE (práh 0,35)	0,000 000 954	0,872	$p < 0{,}01$
NINA	SMALL (práh 0,35)	0,000 000 954	0,872	$p < 0{,}01$
NINA	BASE (práh 0,35)	0,000 000 954	0,872	$p < 0{,}01$
SMALL	BASE (práh 0,35)	0,000 000 954	0,872	$p < 0{,}01$

Tabulka 3: Tabulka výsledků levostranného Wilcoxonova sign rank testu F_1

Detektor	detektor	p-hodnota	p-hodnota po Bonferroniho korekci:
ORG	NINA	0,000 004 07	$p < 0{,}01$
ORG	SMALL (práh 0,5)	0,000 000 319	$p < 0{,}01$
ORG	BASE (práh 0,5)	0,000 000 339	$p < 0{,}01$
NINA	SMALL (práh 0,5)	0,000 000 145	$p < 0{,}01$
NINA	BASE (práh 0,5)	0,000 000 440	$p < 0{,}01$
SMALL	BASE (práh 0,5)	0,001 40	$p < 0{,}01$
ORG	NINA (práh 0,35)	0,000 004 07	$p < 0{,}01$
ORG	SMALL (práh 0,35)	0,000 000 352	$p < 0{,}01$
ORG	BASE (práh 0,35)	0,000 000 352	$p < 0{,}01$
NINA	SMALL (práh 0,35)	0,000 000 296	$p < 0{,}01$
NINA	BASE (práh 0,35)	0,000 000 611	$p < 0{,}01$
SMALL	BASE (práh 0,35)	0,000 911	$p < 0{,}01$

Tabulka 4: Tabulka výsledků levostranného párového t-testu F_β

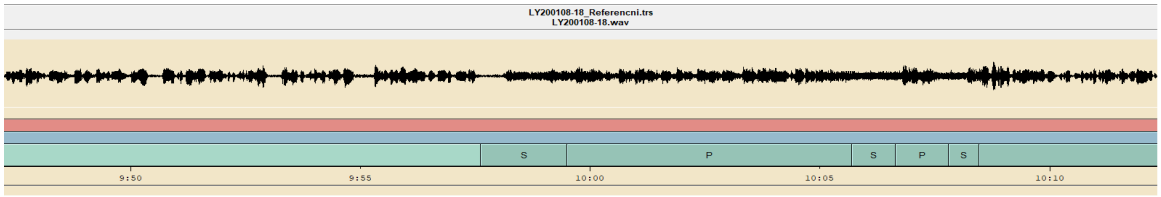
Detektor	detektor	p-hodnota	effect size	p-hodnota po Bonferroniho korekci
ORG	NINA (práh 0,5)	0,000 000 954	0,872	$p < 0{,}01$
ORG	SMALL(práh 0,5)	0,000 000 954	0,872	$p < 0{,}01$
ORG	BASE (práh 0,5)	0,000 000 954	0,872	$p < 0{,}01$
NINA	SMALL (práh 0,5)	0,000 000 954	0,872	$p < 0{,}01$
NINA	BASE (práh 0,5)	0,000 000 954	0,872	$p < 0{,}01$
SMALL	BASE (práh 0,5)	0,000 000 954	0,872	$p < 0{,}01$
ORG	NINA (práh 0,35)	0,000 000 954	0,872	$p < 0{,}01$
ORG	SMALL(práh 0,35)	0,000 000 954	0,872	$p < 0{,}01$
ORG	BASE (práh 0,35)	0,000 000 954	0,872	$p < 0{,}01$
NINA	SMALL (práh 0,35)	0,000 000 954	0,872	$p < 0{,}01$
NINA	BASE (práh 0,35)	0,000 000 954	0,872	$p < 0{,}01$
SMALL	BASE (práh 0,35)	0,000 000 954	0,872	$p < 0{,}01$

Tabulka 5: Tabulka výsledků levostranného Wilcoxonova sign rank testu F_β

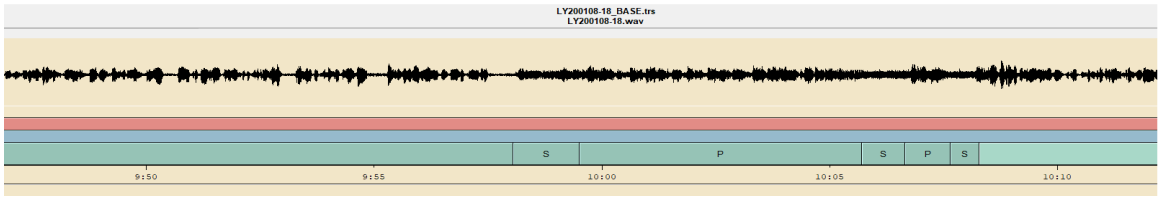
V uvedených tabulkách 3, 4 a 5 jsou u detektorů SMALL a BASE nastavitelné jejich prahy na hodnoty 0,35 nebo 0,5. Práh 0,5 je vhodný pro míru F_1, která uvažuje chyby prvního druhu (detekce neřeči, když ve skutečnosti se jedná o řeč) a druhého druhu (detektor detekuje řeč, ale ve skutečnosti se jedná o neřeč) jako stejně závažné. Práh T menší jak 0,5 pak detektor nastavuje tak, aby teoreticky dosahoval poměru chyb prvního a druhého druhu $T/(1-T)$ (detektor označuje vše, co je vyšší než T, jako řeč). V praxi však je třeba nastavit práh experimentálně.

Z uvedených výsledků vyplývá, že s vysokou (statistickou) významností (chyba prvního druhu hluboko pod 0,05) a bez vyžadování podmínky normality dat F_1 a F_β je: ORG je horší než NINA, NINA je horší než SMALL, SMALL je horší než BASE a samozřejmě z toho plynoucí důsledky, že ORG je horší než SMALL a horší než BASE a NINA horší než BASE). Hodnota efekt size je vyšší než 0,8, což znamená vysokou validitu testu i při datech menšího rozsahu. Můžeme se ptát, co výsledky provedených testů prakticky znamenají a zda je rozdíl mezi detektory prakticky patrný? Odpověď je ano. Na následujících obrázcích je typický příklad srovnání referenčního anotovaného audiosignálu (viz obr. 12) s výstupem BASE detektoru (viz obr. 13), s výstupem detektoru SMALL (viz obr. 14) a konečně i s výstupem stávajícího řešení ORG založeným na CNN (viz obr. 15).

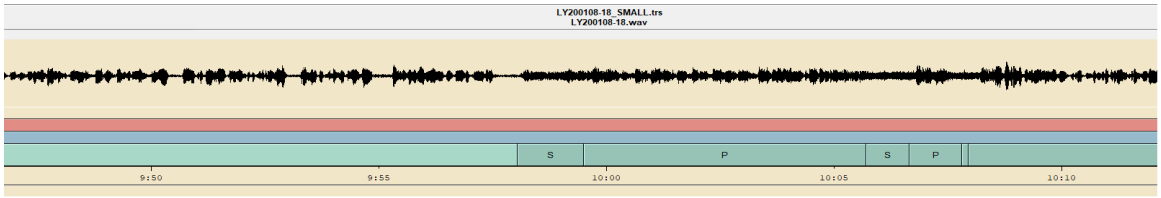
Ukázky výstupu detektorů a referenčního zpracování:



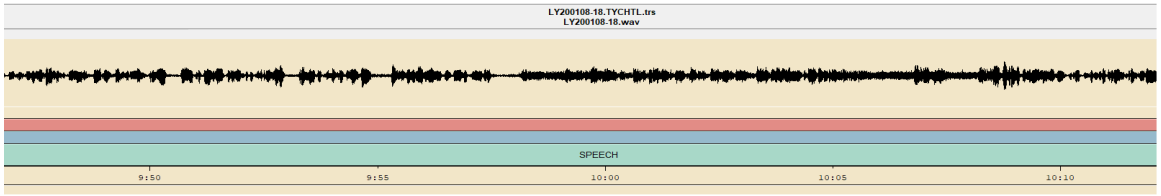
Obrázek 12: Referenční signál



Obrázek 13: výstup detektoru Base



Obrázek 14: výstup detektoru SMALL



Obrázek 15: výstup stávajícího detektoru v českém rozhlase

8 Návrh nového detektoru

Výsledek provedené dosud popsané analýzy (porovnání jednotlivých detektorů) je následující:

1. Všechny navržené detektory (tj. NINA, SMALL, BASE) jsou přesnější než stávající detektor ORG a až na detektor BASE i rychlejší. Má tedy smysl nahradit ORG detektor nějakým nově zkoumaným. Poznamenejme ovšem, že nasazení nového detektoru do prostředí infrastruktury Českého rozhlasu není jednoduchá záležitost (bezpečnostní opatření na straně IT, práva přístupu, firewall, OS Windows server, pokud můžeme vybrat nejdůležitější) a že má smysl o tomto úkonu uvažovat jen za podmínky, že výsledek se projeví výrazným zlepšením stávajícího stavu.

2. Nemá smysl se zabývat nasazením detektoru NINA (je téměř stejně pomalý jako SMALL detektor, ale dosahuje výrazně menší přesnosti).

3. Detektor BASE je výrazně lepší než detektor SMALL, ale je výrazně pomalejší!

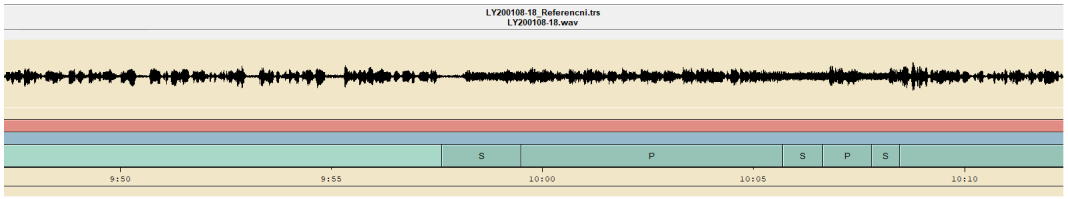
Protože detektor BASE nesplňuje podmínku kladenou na požadovanou rychlost detektoru, mohla by být správná volba použít SMALL detektor. Nicméně jsem se pokusil nalézt lepší řešení, tj. při respektování podmínky požadované rychlosti dosáhnout vyšší přesnosti, než umožňuje detektor SMALL. Základní myšlenka spočívá ve využití kombinace BASE detektoru ke zvýšení přesnosti, a SMALL detektoru k dodržení požadované rychlosti. Dosáhnout toho lze následujícím způsobem: SMALL detektor nechat vyhodnocovat části audiosignálu, které je schopen klasifikovat s vysokou jistotou. BASE detektor pak nechat vyhodnotit jen ty části, které SMALL detektor nezvládne.

Aby byla tato myšlenka schopná realizace, je potřeba mít k dispozici klasifikátor, který rozhoduje, který vstupní úsek signálu SMALL detektor určí správně a který hůře než BASE detektor. Tento klasifikátor (nazvěme jej DISTRIBUTOR) by mohl být realizován jako třetí komponenta celého komplexního detektoru (1. komponenta = SMALL detektor, 2. komponenta = BASE detektor, 3. komponenta = DISTRIBUTOR).

Nabízí se možnost postavit takovýto klasifikátor jako další (hluboký) neuronový model (DNN). Požadavkem ovšem je, aby byl velmi rychlý. Takovýmto klasifikátorem by ale mohl být i samotný SMALL detektor: Jeho výstupem jsou totiž hodnoty logits přiřazené ke každé dvacetině vteřiny vstupního audiosignálu a které zároveň představují důvěru SMALL detektoru v to, že tato dvacetina vteřiny je řeč. SMALL detektor byl natrénován takovým způsobem, že minimalizuje chybu špatné klasifikace řeč/neřeč a zároveň ohodnocuje chybu svého rozhodnutí vahou danou hodnotami logits. Hodnoty logits lze převést na pravděpodobnosti a ty je možno využít pro klasifikaci, zda použít SMALL detektor (tedy již jím provedené výsledky brát jako definitivní), nebo použít BASE detektor, což ale jen znamená jej pro daný úsek signálu spustit. Není tedy potřeba trénovat další klasifikátor, který by navíc zatěžoval výpočetní zdroje. Míra důvěry SMALL detektoru v jeho rozhodnutí je dána hodnotami jeho výstupních pravděpodobností, že jím klasifikovaná část audia je řeč či není řeč. Čím více jsou hodnoty této pravděpodobnosti blízké nule či jedničce, tím více si je detektor jist. Pokud tedy necháme SMALL detektor definitivně rozhodnout, zda jím vyhodnocený pětivteřinový úsek signálu je řeč či řeč není, a pokud pro tento úsek signálu SMALL detektor poskytne na svém výstupu pro těchto 5 vteřin 100 výstupních hodnot pravděpodobností (každou přiřazenou k jedné dvacetině vteřiny) takových, že všechny jsou buď nižší nebo vyšší než určitý práh (např. 0,3 a 0,7), je si SMALL detektor dost jistý svým rozhodnutím a není třeba jeho výsledek revidovat. V opačném případě spustíme na vyhodnocovaném úseku audiosignálu BASE detektor a jeho výstupem pak nahradíme výsledek SMALL detektoru. Celkový čas, za který oba modely vyhodnotí zpracovávaný pětivteřinový úsek audia, je na čtyřjádrovém CPU kratší, než je reálný čas (viz hodnoty časů v tabulce 1). Latence, tj. časové zpoždění detekce za reálným časem vysílání zpracovávaného pětivteřinového úseku audia, by tedy měla být menší než patnáct vteřin (pět vteřin je délka zpracovávaného úseku, pět vteřin jeho pravý kontext, pět vteřin pak trvá vlastní zpracování). Servery Českého rozhlasu mají dva procesory, každý procesor s osmi fyzickými jádry. Podmínka kladená na maximální přípustnou velikost latence cílového řešení (třicet vteřin) by tedy měla být splněna.

9 Nastavení hyperparametrů detektoru

Hyperparametry detektoru (který bude dále v textu nazýván COMBI, protože je ve své podstatě tvořen kombinací SMALL a BASE detektoru) zde představují 3 hodnoty prahů: 2 prahy určující, zda SMALL detektor předá zpracování vstupu BASE detektoru a jeden práh BASE detektoru, nastavené tak, aby míra F_β dosahovala svého maxima. Jak ovšem určit tyto prahy (meze pravděpodobností)? K jejich nastavení je potřeba nové nastavovací datové sady DEVELOPMENT_DATA, která musí být dostatečně obsáhlá a reprezentativní. Tato datová sada proto byla připravena jako další sada, čítá 100 hodin anotovaného audia. Její tvorba je popsaná v kapitole [6] této práce.

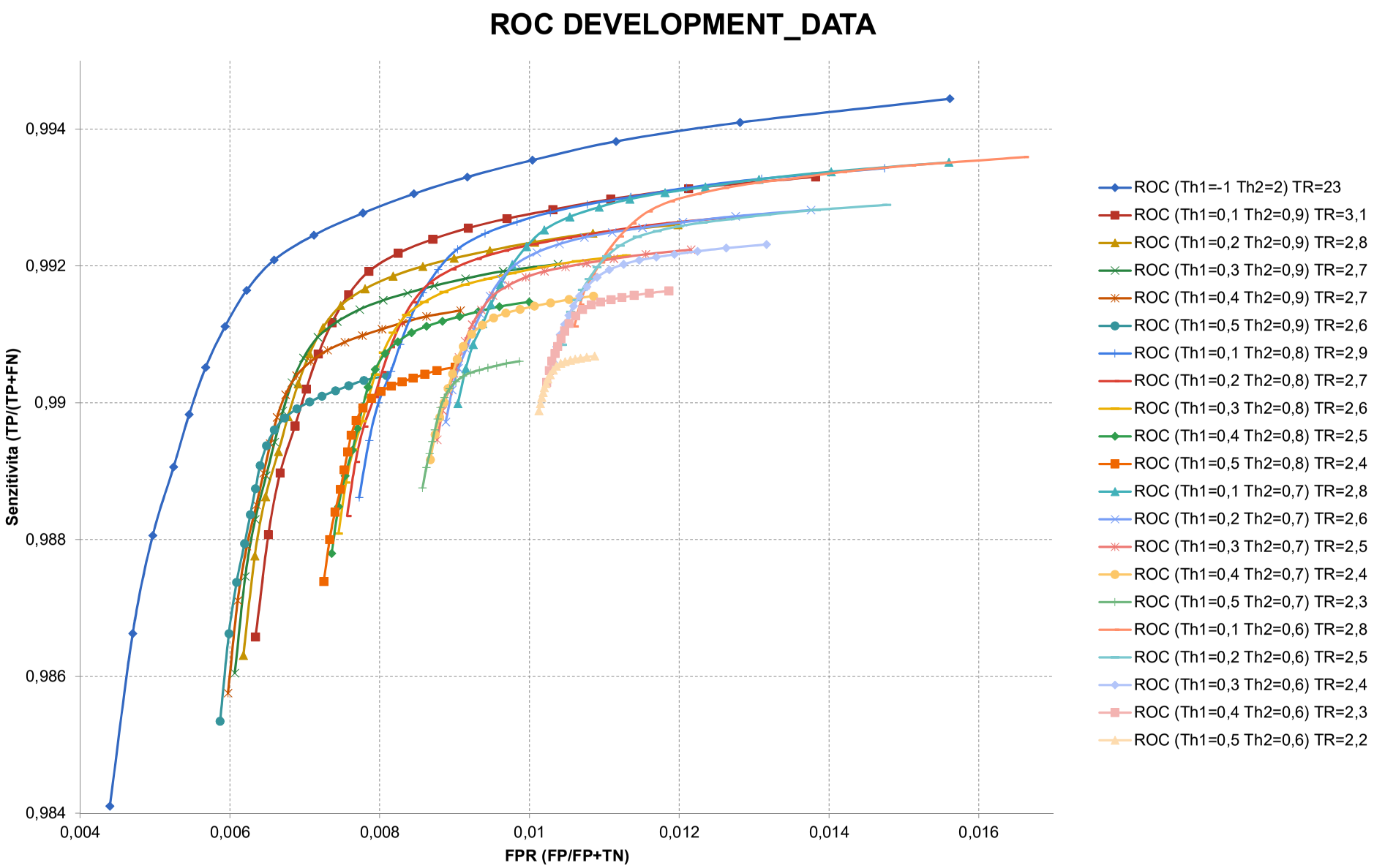


Obrázek 16: Referenční signál

9.1 Nalezení hodnot hyperparametrů

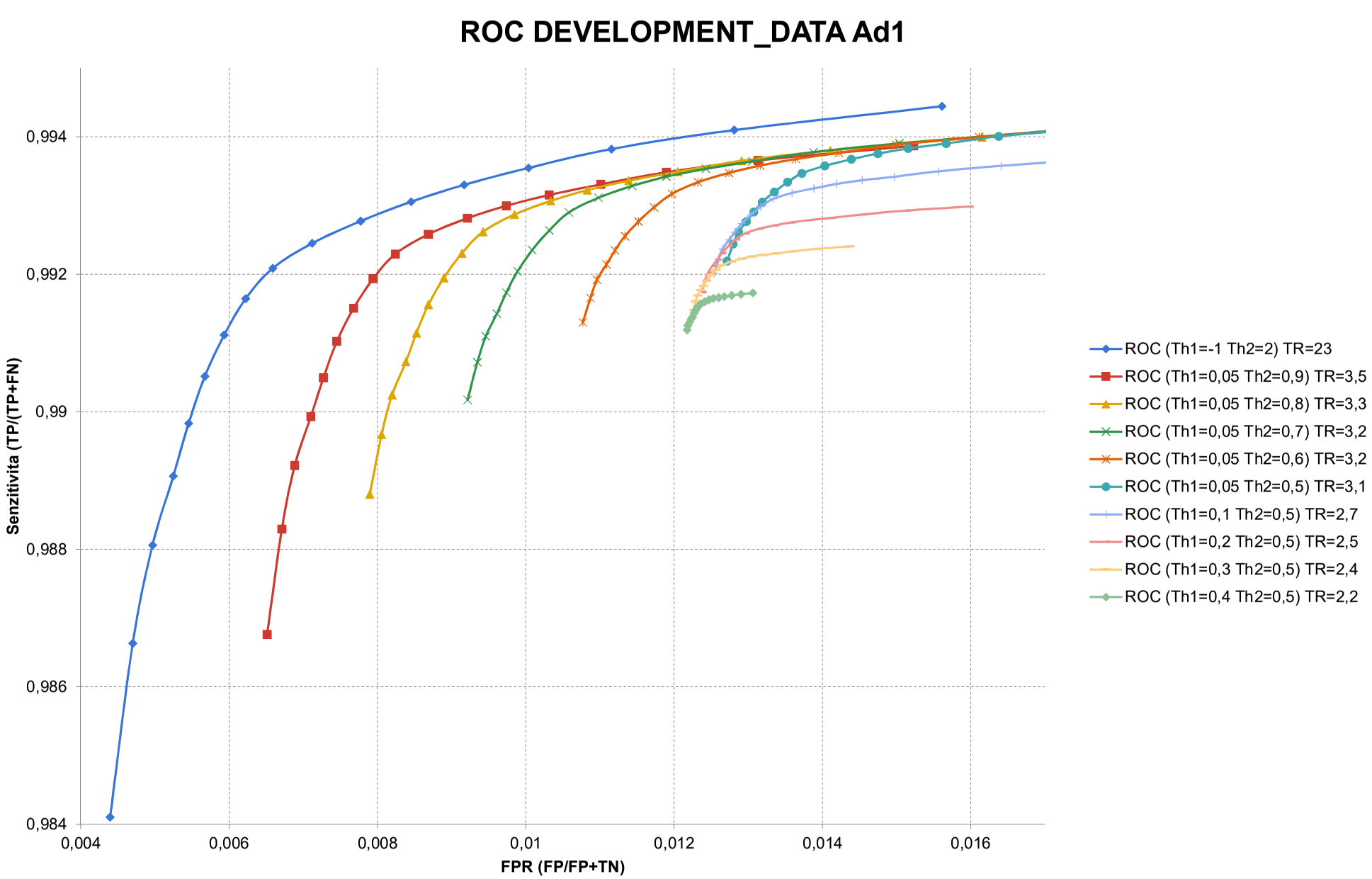
Pro nalezení a nastavení vhodných prahů byly provedeny statisíce testů (kromě vývojové sady byly vyhodnoceny i další sady obsahující celkem více jak 250 hodin referenčního anotovaného audia, každá sada se procházela pro cca 1000 různých nastavení prahů). Na každý test je potřeba mít výstup SMALL a výstup BASE detektoru. Z tohoto důvodu bylo navrženo urychlení: pravděpodobnosti spočítané SMALL a BASE detektorem se ke každéu audiu spočítaly jen jedenkráte (jsou nezávislé na hodnotě prahů) a uloží se. Následný skript pak tyto uložené hodnoty využívá pro vyhodnocení rychlosti a úspěšnosti (počítané mírou F_β) pro různé vstupní konfigurace (hodnoty) prahů. Pro počátek experimentu jsem zvolil základní množinu nastavení prahů $Th1 \in \{0,1, 0,2, ..., 0,5\}$ a $Th2 \in \{0,9, 0,8, 0,7, 0,6\}$. Parametr $Th1$ udává hodnotu nižšího ze dvou prahů SMALL detektoru, $Th2$ pak vyšší hodnotu SMALL detektoru. Po provedení prvního experimentu jsem pro každé nastavení (konfiguraci) prahů SMALL detektoru vykreslil jednu ROC (Receiver Operating Charakteristic), křivku.

Její parametr je hodnota třetího prahu Thb, tj. prahu, podle kterého BASE detektor rozhoduje, zda vstupní úsek klasifikovat jako řeč či neřeč. Výsledné ROC křivky jsou zobrazeny na následujících stranách - viz graf: "ROC DEVELOPMENT_DATA“:

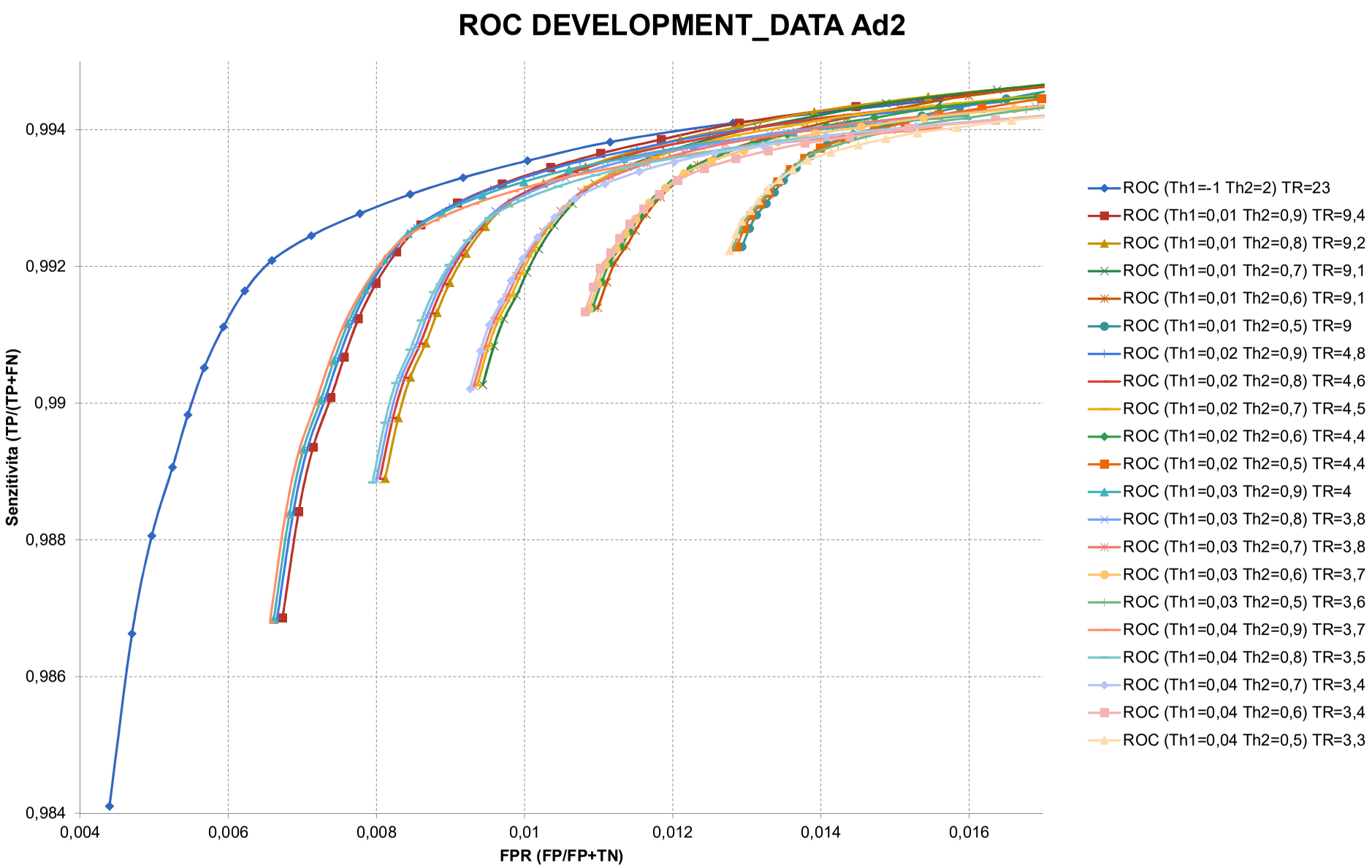

ROC DEVELOPMENT_DATA
Senzitivita (TP/(TP+FN)
0,994
0,992
0,99
0,988
0,986
0,984
0,004
0,006
0,008
0,01
0,012
0,014
0,016
FPR (FP/FP+TN)
ROC (Th1=-1 Th2=2) TR=23
ROC (Th1=0,1 Th2=0,9) TR=3,1
ROC (Th1=0,2 Th2=0,9) TR=2,8
ROC (Th1=0,3 Th2=0,9) TR=2,7
ROC (Th1=0,4 Th2=0,9) TR=2,7
ROC (Th1=0,5 Th2=0,9) TR=2,6
ROC (Th1=0,1 Th2=0,8) TR=2,9
ROC (Th1=0,2 Th2=0,8) TR=2,7
ROC (Th1=0,3 Th2=0,8) TR=2,6
ROC (Th1=0,4 Th2=0,8) TR=2,5
ROC (Th1=0,5 Th2=0,8) TR=2,4
ROC (Th1=0,1 Th2=0,7) TR=2,8
ROC (Th1=0,2 Th2=0,7) TR=2,6
ROC (Th1=0,3 Th2=0,7) TR=2,5
ROC (Th1=0,4 Th2=0,7) TR=2,4
ROC (Th1=0,5 Th2=0,7) TR=2,3
ROC (Th1=0,1 Th2=0,6) TR=2,8
ROC (Th1=0,2 Th2=0,6) TR=2,5
ROC (Th1=0,3 Th2=0,6) TR=2,4
ROC (Th1=0,4 Th2=0,6) TR=2,3
ROC (Th1=0,5 Th2=0,6) TR=2,2

Kromě hodnot prahů $Th1$ a $Th2$ je na obrázcích ROC křivek vypsána pro každou ROC křivku též hodnota TR, která udává poměr času potřebného k provedení výpočtu COMBI detektorem s prahy $Th1$ a $Th2$ ku času potřebnému k provedení výpočtu SMALL detektorem. Pro porovnání ROC křivek s BASE detektorem je na obrázku ROC křivek vynesena modrou barvou i ROC křivka BASE detektoru (jedná se o COMBI detektor s nastavením prahů $Th1$ = -1 a $Th2$ = 2, který je ekvivalentní BASE detektoru).

Z uvedených grafů ROC křivky vyplynulo, že by bylo vhodné provést další experimenty pro hodnoty prahů $Th1 = 0{,}05$ a $Th2 \in \{0,9; 0,8; 0,7; 0,6\}$. Dále pro hodnoty prahů $Th1 \in \{0,1; 0,2; 0,3; 0,4\}$ a $Th2 = 0{,}5$. Tímto tak „vyplnit“ oblast zájmu mezi ROC křivkami nakreslenými na obrázku s prahy $Th1 > 0$ a $Th2 < 1$ a ROC křivkou BASE detektoru. Tento experiment jsem provedl a výsledky opět zobrazil jako ROC křivky „ROC DEVELOPMENT_DATA Ad1“ – viz následující strana.


ROC DEVELOPMENT_DATA Ad1
Senzitivita (TP/(TP+FN)
0,994
0,992
0,99
0,988
0,986
0,984
0,004
0,006
0,008
0,01
0,012
0,014
0,016
FPR (FP/FP+TN)
ROC (Th1=-1 Th2=2) TR=23
ROC (Th1=0,05 Th2=0,9) TR=3,5
ROC (Th1=0,05 Th2=0,8) TR=3,3
ROC (Th1=0,05 Th2=0,7) TR=3,2
ROC (Th1=0,05 Th2=0,6) TR=3,2
ROC (Th1=0,05 Th2=0,5) TR=3,1
ROC (Th1=0,1 Th2=0,5) TR=2,7
ROC (Th1=0,2 Th2=0,5) TR=2,5
ROC (Th1=0,3 Th2=0,5) TR=2,4
ROC (Th1=0,4 Th2=0,5) TR=2,2

Protože výsledky stále ukazovaly vhodnost vyšetřit další části nastavení prahů pro hodnoty prahů $Th1 \in \{0,01; 0,02; 0,03; 0,04\}$ a $Th2 \in \{0,9; 0,8; ...; 0,5\}$, provedl jsem ještě další třetí experiment. Jeho výsledky jsou zobrazeny v podobě ROC křivek „ROC DEVELOPMENT_DATA Ad2“ na následující straně.


ROC DEVELOPMENT_DATA Ad2
Senzitivita (TP/(TP+FN)
0,994
0,992
0,99
0,988
0,986
0,984
0,004
0,006
0,008
0,01
0,012
0,014
0,016
FPR (FP/FP+TN)
ROC (Th1=-1 Th2=2) TR=23
ROC (Th1=0,01 Th2=0,9) TR=9,4
ROC (Th1=0,01 Th2=0,8) TR=9,2
ROC (Th1=0,01 Th2=0,7) TR=9,1
ROC (Th1=0,01 Th2=0,6) TR=9,1
ROC (Th1=0,01 Th2=0,5) TR=9
ROC (Th1=0,02 Th2=0,9) TR=4,8
ROC (Th1=0,02 Th2=0,8) TR=4,6
ROC (Th1=0,02 Th2=0,7) TR=4,5
ROC (Th1=0,02 Th2=0,6) TR=4,4
ROC (Th1=0,02 Th2=0,5) TR=4,4
ROC (Th1=0,03 Th2=0,9) TR=4
ROC (Th1=0,03 Th2=0,8) TR=3,8
ROC (Th1=0,03 Th2=0,7) TR=3,8
ROC (Th1=0,03 Th2=0,6) TR=3,7
ROC (Th1=0,03 Th2=0,5) TR=3,6
ROC (Th1=0,04 Th2=0,9) TR=3,7
ROC (Th1=0,04 Th2=0,8) TR=3,5
ROC (Th1=0,04 Th2=0,7) TR=3,4
ROC (Th1=0,04 Th2=0,6) TR=3,4
ROC (Th1=0,04 Th2=0,5) TR=3,3

Výsledky třetího experimentu však již nepřináší oproti druhému experimentu výrazné zlepšení a čas potřebný k výpočtu již dosahuje kritické hodnoty 1,2 násobku výpočetního času původního ORG detektoru nebo jej dokonce i výrazně převyšuje. Jako nejvhodnější kandidáti pro výslednou implementaci se jeví ROC křivky z množiny „ROC DEVELOPMENT_DATA Ad1“. Výběr nejvhodnější ROC křivky z této množiny ROC křivek musí splňovat podmínku kladenou na rychlost a dosahovat co největší přesnosti. Této podmínce vyhovuje jediná ROC křivka s prahy $Th1 = 0,05$ a $Th2 = 0,7$. Zbývá určit hodnotu na této křivce odpovídající prahu BASE detektoru Thb. Tato hodnota byla vybrána jako hodnota ROC křivky (s prahy $Th1 = 0,05$ a $Th2 = 0,7$), pro niž je hodnota míry F_β maximální. Tato hodnota činí 0,9925 a nastává pro hodnotu prahu $Thb = 0,35$. Výsledné hyperparametry COMBI detektoru tedy jsou: $Th1 = 0,05$, $Th2 = 0,7$ a $Thb = 0,35$. S těmito hodnotami dosahuje COMBI detektor následujících výkonů: Výsledný navržený detektor je v průměru 1,16 pomalejší než stávající řešení ORG při hodnotě míry $F_\beta = 0,9925$ oproti hodnotě míry $F_\beta = 0,8470$ (pro $\beta = 2$) detektoru ORG. Oproti detektoru SMALL je v průměru 3,2krát pomalejší (ale stále s rezervou v mezích požadované rychlosti) a přitom dosahuje oproti SMALL detektoru lepší hodnoty míry F_β, která je u SMALL detektoru rovna 0,9906).

Pro lepší představu představuje rozdíl v hodnotě F_β mezi SMALL detektorem a optimálně nastaveným COMBI detektorem snížení relativní chyby prvního druhu o 31 % (relativně vzhledem k chybě SMALL detektoru, tj. chyba SMALL detektoru představuje 100 %) při současném snížení relativní chyby druhého druhu o 46 %, čili navržený detektor snižuje oproti modelu SMALL chybu detekce řeči téměř o třetinu při téměř poloviční hodnotě chyby detekce neřečových úseků.

V tabulce 6 je uvedeno srovnání měr F_β a F_1 nastaveného COMBI detektoru s detektory ORG, NINA a SMALL s prahem 0,35. Všechny výsledky jsou statisticky významné s p-hodnotou $< 0,01$.

F_β			
Detektor	detektor	t-test	Wilcoxon
ORG	$\text{COMBI}_{Th1=0,05 Th2=0,7 Thb=0,35}$	0,000 000 355	0,000 000 954
NINA	$\text{COMBI}_{Th1=0,05 Th2=0,7 Thb=0,35}$	0,000 000 452	0,000 000 954
$\text{SMALL}_{Th=0,35}$	$\text{COMBI}_{Th1=0,05 Th2=0,7 Thb=0,35}$	0,001 484	0,000 000 954

Tabulka 6: Srovnání detektorů podle míry F_β

F_1			
Detektor	detektor	t-test	Wilcoxon
ORG	$\text{COMBI}_{Th1=0,05 Th2=0,7 Thb=0,35}$	0,000 000 092	0,000 000 954
NINA	$\text{COMBI}_{Th1=0,05 Th2=0,7 Thb=0,35}$	0,000 000 806	0,000 000 954
$\text{SMALL}_{Th=0,35}$	$\text{COMBI}_{Th1=0,05 Th2=0,7 Thb=0,35}$	0,002	0,000 000 954

Tabulka 7: Srovnání detektorů podle míry F_1

Legenda k tabulkám 6 a 7:

t-test ... p-hodnota levostranného párového t-testu H_0: detektor vlevo není horší než detektor vpravo).

Wilcoxon ... p-hodnota levostranného Wilcoxonova sign rank testu (H_0: detektor vlevo není horší než detektor vpravo).

Oproti BASE detektoru pracuje nastavený COMBI detektor v průměru více jak 7-krát rychleji. Oproti stávajícímu detektoru ORG pak nový detektor snižuje relativní chybu prvního druhu o 98 % a relativní chybu druhého druhu pak o 89 %! Srovnání detektorů z pohledu četností jejich chyb nebo z pohledu jejich absolutních chyb uvádí tabulky 8 a 9.

	TP	FN	FP	TN
ORG	31413,28	5778,61	2377,07	32809,74
NINA	35675,13	1494,23	282,48	34876,75
$\text{SMALL}_{Th=0,35}$	37009,22	191,68	491,61	34620
$\text{BASE}_{Th=0,35}$	37084,87	121,4	140,66	35028,76
$\text{COMBI}_{Th1=0,05Th2=0,7Thb=0,35}$	37073,79	132,19	266,4	34897,53

Tabulka 8: Srovnání detektorů podle počtu jejich chyb

	$FPR = FP/(FP+TN)$ (Chyba I. druhu)	$FNR = FN/(FN+TP)$ (Chyba II. druhu)
ORG	0,155	0,068
NINA	0,040	0,008
$\text{SMALL}_{Th=0,35}$	0,0052	0,014
$\text{BASE}_{Th=0,35}$	0,0033	0,004
$\text{COMBI}_{Th1=0,05Th2=0,7Thb=0,35}$	0,0036	0,0076

Tabulka 9: Srovnání detektorů podle jejich absolutních chyb

Vysvětlení symbolů je uvedeno níže:

TP ... True Positives

FN ... False Negatives

FP ... False Positives

TN ... True Negatives

FPR ... False Positive Rate = 1 - specificity

FNR ... False Negative Rate = 1 - sensitivity

Th ... rozhodovací práh (IF prob modelu $> Th$ THEN řeč ELSE neřeč)

$Th1$ ... 1. rozhodovací práh COMBI modelu

$Th2$ ... 2. rozhodovací práh COMBI modelu (IF prob SMALL modelu $> Th1$ AND prob SMALL modelu $< Th2$ THEN nech rozhodnout BASE model ELSE rozhodni sám)

Thb ... 3. rozhodovací práh COMBI modelu (IF prob BASE modelu $> Thb$ THEN řeč ELSE neřeč).

Úspěšnost výsledného nastavení prahů (vyjádřené mírou F_{beta} jsem se pokusil porovnat s úspěšností (mírou F_{beta}) "sousedních" nastavení prahů, tj. s konfiguracemi prahů ležícími na sousedních ROC křivkách či na ROC křivce s nalezenými optimálními hodnotami.

Výsledky provedených testů EVAL_DATA datové sadě potvrdily, že hod-

noty hyperparametrů COMBI detektoru:

$$Th1 = 0,05,\ Th2 = 0,7 \text{ a } Thb = 0,35$$

dávají statisticky významně lepší výsledky než sousední „neoptimální“ hodnoty:

$$Th1 = 0,05,\ Th2 = 0,6 \text{ a } Thb = 0,35.$$

(párový jednostranný t-test: p-hodnota $<$ 0,002, jednostranný Wilcoxonův sign rank test: p-hodnota $<$ 0,002).

Test srovnání hodnot hyperparametrů COMBI detektoru:

$$Th1 = 0,05,\ Th2 = 0,7 \text{ a } Thb = 0,35$$

s hodnotami

$$Th1 = 0,1,\ Th2 = 0,7 \text{ a } Thb = 0,35.$$

nepotvrdil, že nastavení $Th1 = 0,05$, $Th2 = 0,7$ a $Thb = 0,35$ dává statisticky významně lepší výsledky než uvedené sousední hodnoty. Tento výsledek je nejspíše způsoben malou hodnotou rozsahu náhodného výběru, která je dána velikostí použité datové sady a nestačí k prokázání odlišnosti dvou nastavení lišících se o malou hodnotu (0,05). COMBI detektor nastavený na hodnoty $Th1 = 0,05$, $Th2 = 0,7$ a $Thb = 0,35$ by ale teoreticky měl být lepší, protože přenechává BASE modelu k rozhodnuti větší část audiosignálu než detektor s nastavením prahů $Th1 = 0,1$, $Th2 = 0,7$ a $Thb = 0,35$.

Pro zbylé sousední konfigurace:

$$Th1 = 0,05,\ Th2 = 0,8 \text{ a } Thb = 0,35\ ,$$
$$Th1 = 0,04,\ Th2 = 0,7 \text{ a } Thb = 0,35$$

dává již COMBI detektor příliš vysoké hodnoty výpočetního času.

9.2 Realizace nového detektoru

V přechozí části jsem popsal postup nalezení cílových hodnot hyperparametrů modelu COMBI postaveného na dvou wav2vec 2.0 modelech, pomalejším BASE a rychlejším ale méně přesném SMALL modelu. Optimální hodnoty parametrů pro detektor činí $Th1 = 0,05$, $Th2 = 0,7$, $Thb = 0,35$. Výsledný

COMBI detektor byl s těmito hodnotami hyperparametrů implementován v jazyce Python s podporou frameworku Pytorch a dalších knihoven. Dále byl upraven tak, aby vracel identifikátor, zda je ve vstupním signálu řeč či ne, ke každé desetině sekundy vstupního audiosouboru. Vstupem detektoru je 16 kHz audio délky 15 sekund přijímané každých 5 sekund. Model pak každých 5 sekund vstupní navzorkovaný signál délky 15 sekund zpracuje a pro prostředních 5 sekund vrátí každých 5 sekund výsledek v podobě posloupnosti délky 50 čísel obsahujících 0 nebo 1. Detektor tedy ve svém výstupu vrací každých 5 sekund 50 booleovských hodnot. Hodnota 1 značí, že v desetině sekundy časově odpovídající výstupní hodnotě 1 je řečový signál, v opačném případě není řeč v této desetině sekundy vstupního audia obsažena. Detektoru s touto specifikací byl předán k vyzkoušení pro následnou implementaci.

10 Zhodnocení a závěr

Cílem práce bylo navrhnout řešení, které by napomohlo řešit úlohu automatického vystavení vybraných pořadů Českého rozhlasu na internetovém portálu mujRozhlas.cz. Pro úspěšné zvládnutí tohoto úkolu je nutné automaticky poměrně přesně určit začátek a konec vystavovaného pořadu, tak aby pořad nebyl na svém začátku ani konci ustřižen a na druhou stranu neobsahoval části okolních pořadů. To je důležité především pro případy, kdy publikování okolních pořadů nebo i jen jejich částí je vázáno licenčními podmínkami, jako tomu je např. v případě hudební produkce, kdy neoprávněné vystavení částí hudební skladby na veřejném internetovém portálu by mohlo vést k právní žalobě podané na vystavovatele pro porušení autorského zákona. Systém automatického vystřihování pořadů musí být proto robustní a přesný. Jeho architektura je velmi komplexní a sestává z mnoha podpůrných částí. Jednou z klíčových komponent tohoto systému je část řešící detekci řečových a ostatních neřečových úseků signálu. Stávající software detekce řeči a neřeči užívané v Českém rozhlasu zatím neposkytuje potřebnou přesnost. Cílem mé práce bylo proto pokusit se tento stav zlepšit.

V této práci jsem vybral několik perspektivních metod detekce řeči v audiosignálu, změřil jejich přesnost a výpočetní náročnost, provedl jejich vzájemné porovnání a jejich srovnání se stávající metodou implementovanou v Českém rozhlase. Pro provedení porovnávacích testů bylo třeba připravit datové sady z řešené domény, tj. opatřit informaci k mnoha hodinám vysílání, kdy se ve vysílaném audiu hovoří a kdy se jedná o jiné než řečové úseky. Přestože s počátku na úloze pracovalo několik anotátorů, jejich výkon nebyl dle kladených požadavků na anotace zcela uspokojivý, a i z důvodu mezianotátorské variability (rozdílného značení téhož audioúseku různými anotátory) jsem podstatnou část přípravy datových sad zajistil sám, Výhodou je poměrně jednotná anotace všech zvukových nahrávek v datových sadách a získání detailního přehledu o zvukových jevech vyskytujících se během vysílání.

Po vyhodnocení vybraných metod na první evaluační vytvořené datové sadě se jako nejperspektivnější jevily dva modely, oba postaveny na přístupu wav2vec 2.0. Jeden model splňoval požadovanou rychlost výpočtu, ale byl méně

přesný než model druhý, jehož výpočetní náročnost však mnohonásobně převyšovala maximálně povolenou mez. Navrhl jsem proto novou architekturu detektoru postavenou na kombinaci těchto dvou modelů. Tuto navrženou architekturu jsem odzkoušel a nastavil její parametry (hyperparametry dílčích modelů). K nastavení hodnot hyperparametrů jsem použil novou rozsáhlou datovou sadu, již jsem vytvořil. Pro nastavení nového detektoru řeči bylo nutné provést poměrně rozsáhlé experimenty (zpracován ekvivalent půl milionu hodin audia). Pro účely vyhodnocení a nastavení byla v průběhu práce vytvořena řada skriptů (psaných v jazyce Python), které mohou být užitečné pro řešení dalších úloh.

Přidanou hodnotou této práce je tedy splnění zadání diplomové práce, kdy jsem zanalyzoval metody možného řešení, porovnal je se stávajícím řešením i mezi sebou navzájem. Dále jsem vytvořil datové sady a řadu skriptů pro porovnávání různých metod i pro vyhodnocení nového možného řešení. Na základě provedené analýzy a porovnání jednotlivých metod jsem pak navrhl a realizoval řešení, kdy výsledný detektor je pouze 1,16-krát pomalejší než detektor původní s poklesem absolutní chyby prvního druhu (neidentifikovaná řeč) na EVAL_DATA datové sadě z 15,5 % na 0,36 % a s poklesem absolutní chyby druhého druhu (neidentifikovaná neřeč) ze 7 % na 0,8 %. Oproti původnímu detektoru ORG tedy nový detektor snižuje relativní chybu prvního druhu o 98 % a relativní chybu druhého druhu o 89 %.

V současné době se řeší implementační práce nového detektoru v infrastruktuře Analytiky vysílání Českého rozhlasu a zároveň probíhají práce na přetrénování modelů na datech Českého rozhlasu. Budu velmi rád, pokud má práce pomůže přispět k úspěšnému vyřešení úlohy automatické detekce začátků a konců pořadů pro bezproblémové vystavování pořadů na internetovém portálu mujRozhlas.cz.

Literatura

[1] Alexei Baevski et al. „wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations“. In: *Advances in Neural Information Processing Systems*. Ed. H. Larochelle et al. Sv. 33. Curran Associates, Inc., 2020, s. 12449–12460. URL: https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf.

[2] Claude Barras et al. „Transcriber: a free tool for segmenting, labeling and transcribing speech.“ In: *LREC*. Sv. 98. 1998, s. 28–30.

[3] David Doukhan et al. „An open-source speaker gender detection framework for monitoring gender equality“. In: *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2018, s. 5214–5218.

[4] David Doukhan et al. „Describing Gender Equality in French Audiovisual Streams with a Deep Learning Approach“. In: *VIEW Journal of European Television History and Culture* 7 (pros. 2018), s. 103–122. DOI: 10.18146/2213-0969.2018.jethc156.

[5] David Doukhan et al. „Ina's Mirex 2018 music and speech detection system. In Music Information Retrieval Evaluation eXchange (MIREX 2018).“ In: zář. 2018, s. 2.

[6] Haytham Fayek. „Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between [Online] Available at: haythamfayek. com“. In: (2016). URL: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html.

[7] Yu-An Chung et al. „w2v-BERT: Combining Contrastive Learning and Masked Language Modeling for Self-Supervised Speech Pre-Training“. In: *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2021, s. 244–250. DOI: 10.1109/ASRU51503.2021.9688253.

[8] Abdelrahman Mohamed, Dmytro Okhonko a Luke Zettlemoyer. „Transformers with convolutional context for ASR“. In: *CoRR* abs/1904.11660 (2019). arXiv: 1904.11660. URL: http://arxiv.org/abs/1904.11660.

[9] Eric D. Scheirer a Malcolm Slaney. „Construction and evaluation of a robust multifeature speech/music discriminator“. In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing* 2 (1997), 1331–1334 vol.2.

[10] David Snyder, Guoguo Chen a Daniel Povey. *MUSAN: A Music, Speech, and Noise Corpus*. 2015. DOI: 10.48550/ARXIV.1510.08484. URL: https://arxiv.org/abs/1510.08484.

[11] George Tzanetakis a Perry Cook. „Musical genreclassification of audio signals. IEEE Transactions onspeech and audio processing,“ in: *CoRR* abs/1904.11660 (2002). eprint: 10(5):293302,2002.

[12] Ashish Vaswani et al. „Attention is All you Need“. In: *Advances in Neural Information Processing Systems*. Ed. I. Guyon et al. Sv. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

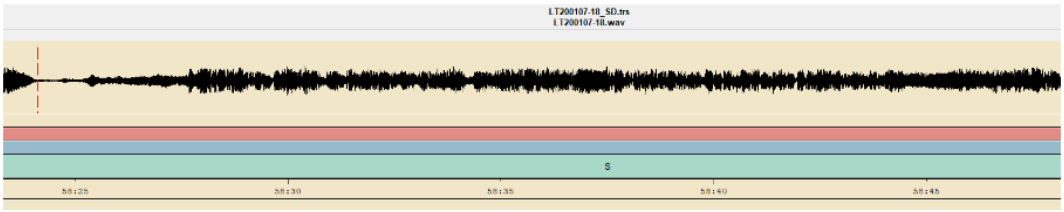
Příloha A

V této příloze se zejména věnujeme konkrétním případům z anotování, aby posloužily tyto případy jako možné vodítko jak pro další zpracování, tak ale také pro rozluštění systému jak celkově model detekuje jednotlivé pasáže jako řeč, či neřeč:

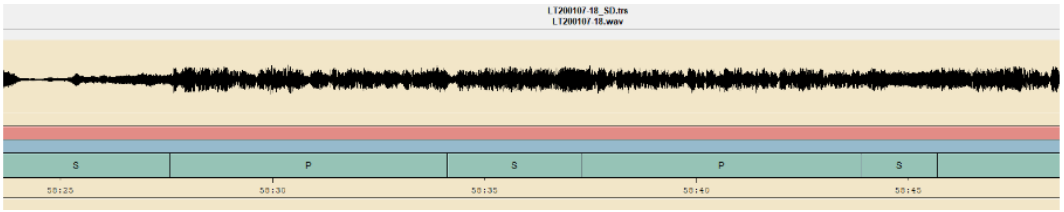
Anotování před a po

Anotované úseky audio dat se vytvářely systému Transcriber 1.5.1 za vstupu audiosignálu a příslušného vygenerovaného souboru .trs. Na obrázcích níže je možné vidět rozčlěnění audiosignálu, i po předzpracování VAD je nutné upravit, posunout, nebo v tomto případě vytvořit nové hranice neřečových a řečových signálů.

Ukázka z programu transcriber před

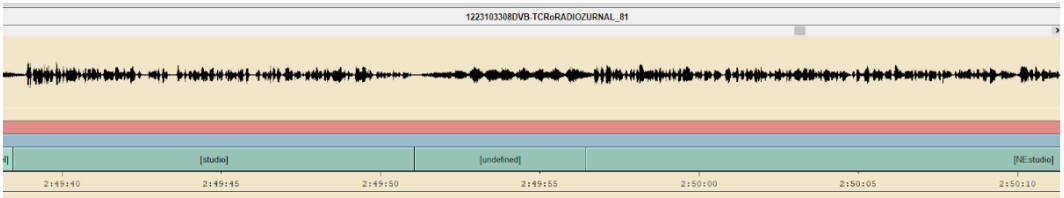


Ukázka z programu transcriber po provedené anotací



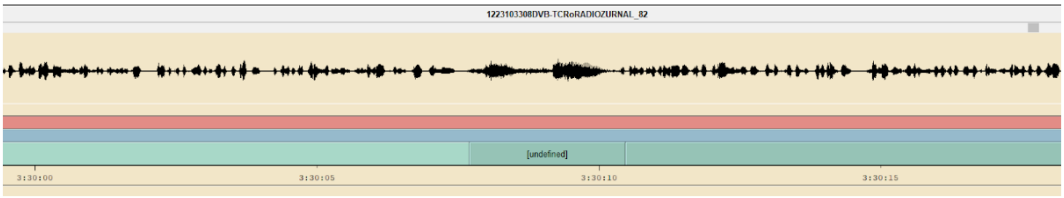
Koleda dětská

Na Stanici Radiozurnal v r. 2019 23.12 v 10:00, V čase 2:49:50 je zpínavá koleda dětmi předěl mezi moderátorem a moderátorem z terénu. Dobré oddělit.

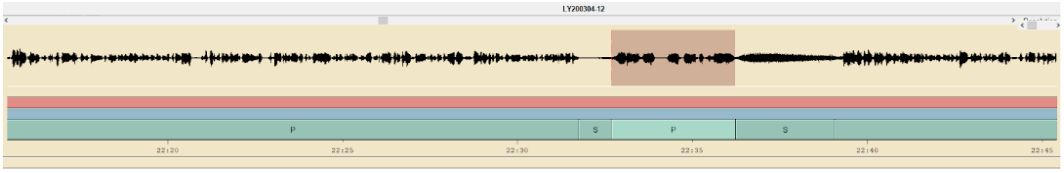


výkřiky moderátora

Chyba v trénovacích datech: 1223103308DVB-TCRoRADIOZURNAL_82



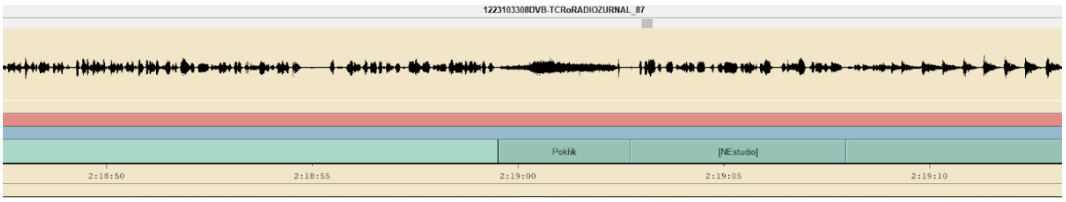
výkřiky řečníka



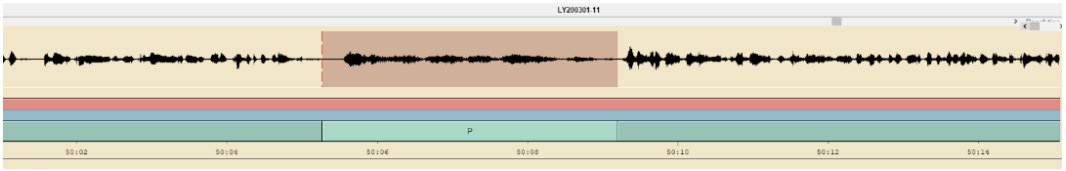
výkřiky fanoušků

1223103308DVB-TCRoRADIOZURNAL 87. Pokřik na podporu sportovce či závodníka či závodnice jsou obecně náročné, mnohdy nelze totiž informaci vyextrahovat, převést na text. Většinou jsou dokonce takovéto úseky na předělech - předání slova od moderátora ze studia do terénu nebo do reportáže. Takovéto úseky, které pak nelze rozklíčovat jsou vnímané z pohledu této práce jako neřečové z toho důvodu, aby systém následně měl lepší pozici při rozlišování pořadů. Nejprve je uvededen příklad ze závodu, Kde je rozeznatelná řeč (pokřik) podpora Olíííí ve druhém pak je skupina fotbalových ultras, kde informace nelze být dekodována a je označena jako neřeč. Obě varianty jsou však z pohledu jednoty vedeny jako předěl mezi studiem a terénem a tedy jsou anotovány jako neřeč.

Pokřik a podporu sportovkyně Olíííí.

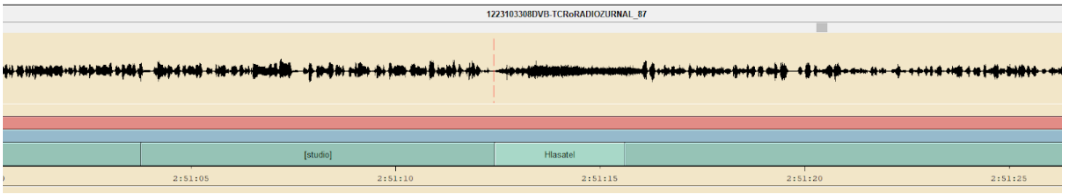


Křik na stadionu - úryvek ČRo Radiožirnál z r. 2020 3.ledna v 11:50:06 (soubor LY200301-11).



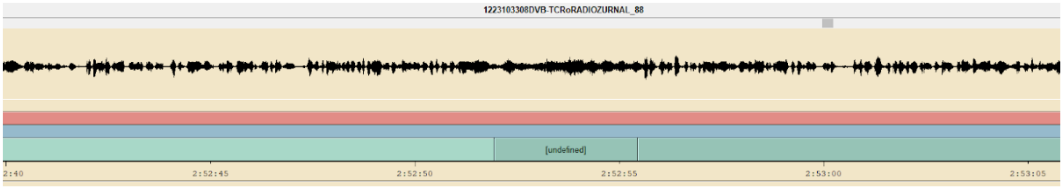
Hlasatel americký jazyk

Zde je popsán vstup amerického hlasatele ze zápasu z terénu - cizí řeč v podkresu šumu a všeobecného hlásotu davu fanoušků, nicméně je zde jasná zakodvaná informace "score"- tedy gol.



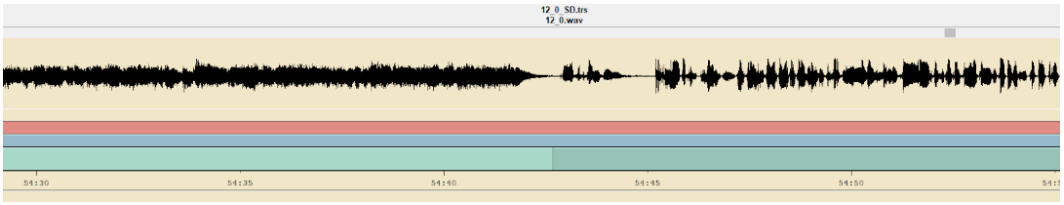
Církevní chorály

Zde je uveden příklad anotování ze vstupu církevní mše, kdy se proměňuje řeč, tedy kázání kněze, jenž následně přechází do zpěvu. Zde je nutné oddělit skutečný text (řeč), která se má rozpoznat s recitativem, který se již rozpoznávat rozpoznávačem nemá - nepřináší žádnou smysluplnou informaci.



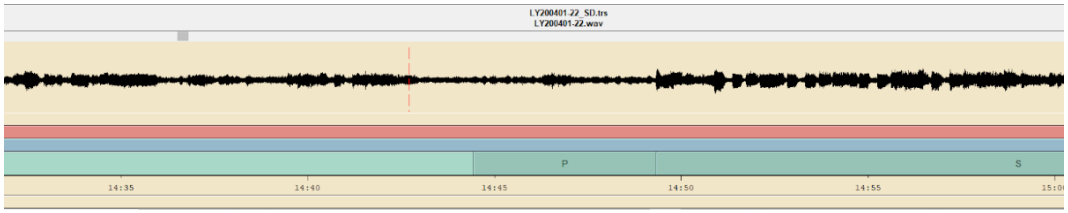
Církevní Kázání

Na Stanici ČRo Brno úryvek z r. 2017 12.listopadu v přibližně 12:54:40 (soubor LB171112-12) se naopak již jedná o kázání kazatele v kostele. Systém při nedostatečném trénování pouze hlasu na písně může celá slova vyhodnotit jako zpěv, byť ještě kazatel pronáší kázání, ale protahuje vokály. Pro potřeby ČRo je kázání potřeba rozpoznávat a předkládat uživatelům dle jejich práv.



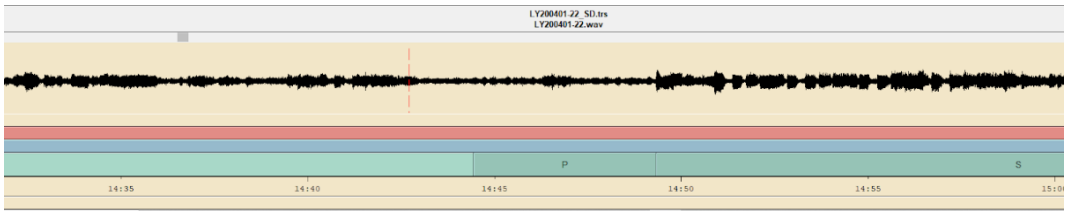
False positive detekování řeči

v souboru *LY200401-22_SD*, tedy ČRo radiožurnál v r. 2020 4.ledna ve 22:14:45 je detekována uprostřed písně řeč.



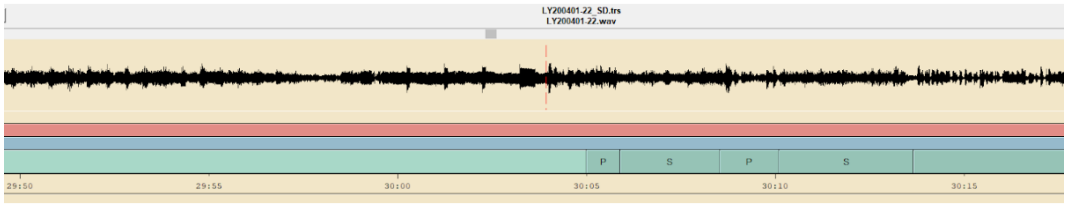
False positive detekování řeči

v souboru *LY200401-22_SD*, tedy ČRo radiožurnál v r. 2020 4.ledna ve 22:14:45 je detekována uprostřed písně řeč.



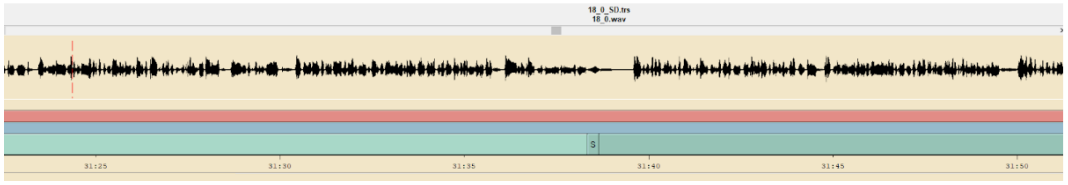
False negative detekování řeči

v souboru *LY200401_22_SD*, ČRo radiožurnál v r. 2020 4.ledna ve 22:30:03 je fals negative, začátek moderátora nebyl zachycen.



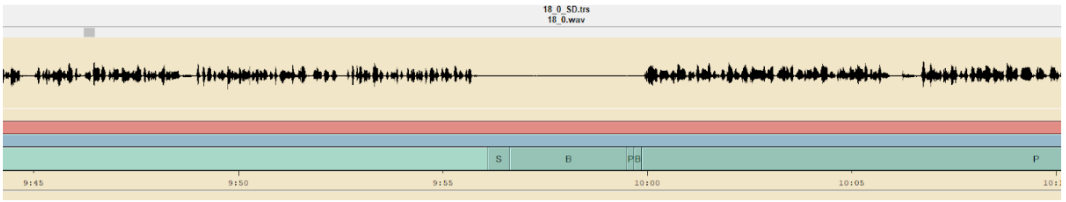
False positive detekování řeči

v souboru *LS171003-18*, ČRo Plus v r. 2017 3.října v 18:31:37 šum, takzvané upípnutí do čistého hlasu moderátora a tím přerušení promluvy.



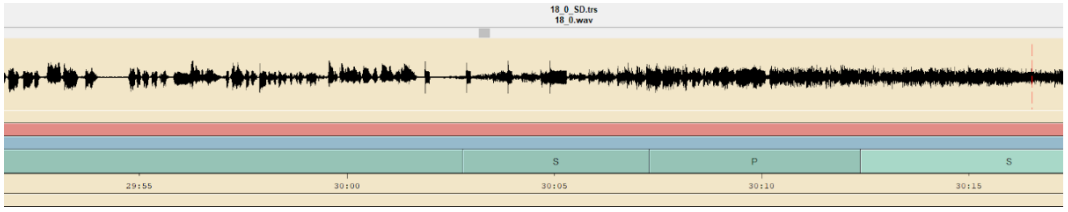
False positive detekování řeči

v souboru *LS171031-18*, ČRo Plus v r. 2017 taktéž 3.října v 18:09:55 je odmlka - ticho. V tomto případ byl dokonce rozpoznávačem rozpoznána promluva, byť zde žádná není. Opraveno na neřeč.



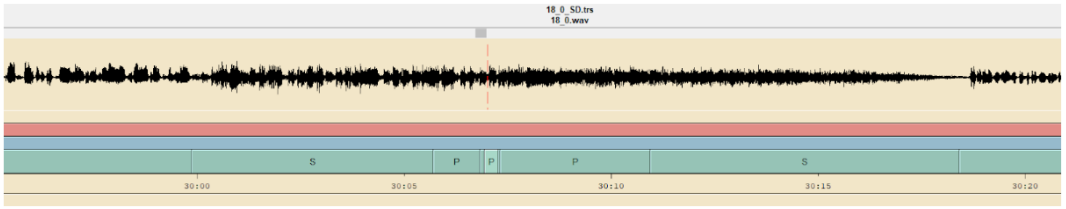
Ztlumený hlas moderátora

Tato řeč je velmi potlačena, tento úryvek sic nepochází ze stanice ČRo Plus, nicméně je důležitý z hlediska nasazování na ostatních stanicích, kde je takovýto jev, zvláště pak v nočních hodinách poměrně častý. V souboru *LS171031-18*, ČRo Plus v r. 2017 31.října v 18:30:15 je velmi ztlumený hlas, nicméně informace je zde také možná vydekodovat, proto se takovýto úsek a jemu podobné anotují jako řeč.



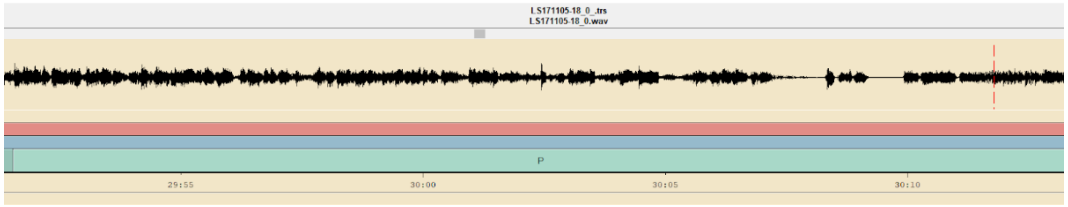
Moderátorem vstup čísel

Přeřeknutí moderátora v souboru *LS171101-18*, tedy ČRo Plus 2017 17.listopadu v 18:30:10 je přeřeknutí, informace je zde ale patrná. Rozpoznávač by měl tento úsek následně předat k rozpoznání obsahu a převodu na text.



Technická chyba, kolize reklamy a časomíry

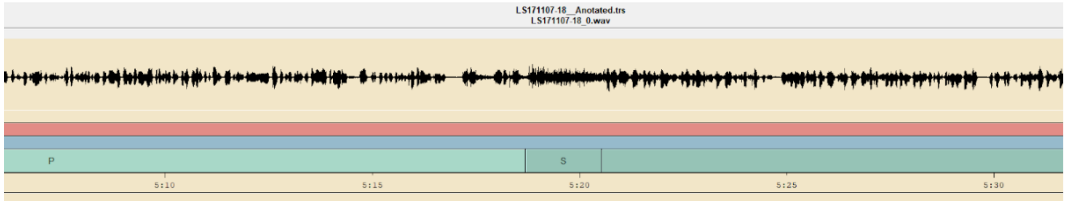
V tomto úryvku se jedná o velmi potlačenou řeč, nicméně v tomto případě se jedná o úryvek reklamy. Reklamy jsou velice specifické úseky, kde je řeč uměle předělána (remixována) s uměle vytvořeným podkresem. Pokud tedy navíc v souboru *LS171105-18*, ČRo Plus v r. 2017 17.listopadu v 18:30:10 je opomenuto ztlumení řeči od moderátorů, je pak tedy tento hlas spíše podkresem úvodní znělky pro odpočet před zprávami a časomírou. Běžně by anotátor tento úsek označil za neřeč, nicméně jelikož se stále dá extrahovat informace, chceme, aby detektor aktivity řeči detekoval řeč.



Rušivé úryvky vstupů

Zde se jedná o vstup z videohry. Tyto úseky bývají obtížně detekovatelné a jedná se o jeden z problematických částí, jak pro anotování, tak ale i pro následnou detekci. Jelikož je zde míchána znělka videohry, (v tomto případě

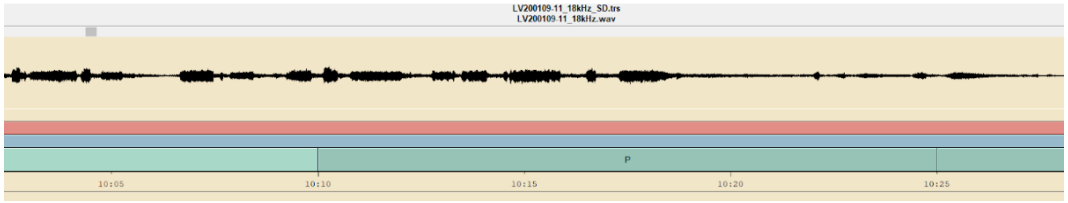
Call-of-Duty), je v úryvku řeč naprosto přehlušena hlukem (výstřely, výbuchy, křik, či ječení). Navíc se k tomu můžou míchat různé hlasy jako je Německý, Anglický nebo dokonce i orientální jazyky. V těchto případech je tedy potřeba selektovat použitelnou řeč, se kterou se mísí parazitní frekvence hluku. V jistých případech je sice možné rušivý šum filtrovat a je dost možné, že se detektory založených wav2vec naleznou charakteristiky řeči (dokáží i přes velmi zašuměný podkres detekovat řeč), avšak v tomto případě je dobré tyto úseky selektovat a označit jako neřeč i vzhledem k ostatním souborům, jako jsou vstupy koncertů, při haváriích a podobně. Proto je v souboru *LS171107-18*, ČRo Plus v r. 2017 7.listopadu v 18:05:20 signál rozdělen a vložen úsek se šumem, anotovaný na neřeč.



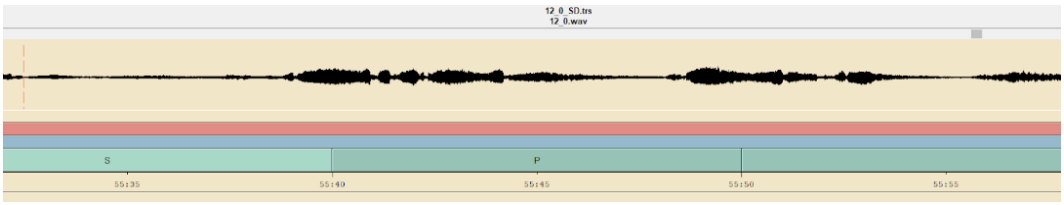
Operní nebo lidový (*dechovkový*) zpěv

Stanice Vltava a Brno mají časté hudební pořady. Zde je velkou výzvou detekovat lidové a operní zpěvy. Anotátor, musí zvážit, jedná-li se o úsek patřící k písni. Ještě obtížnější je situace, kdy je zpěvák do studia a má promluvu. Mezo promluvou a začátkem písně pak není téměř žádný rozdíl. Nicméně je potřeba z pohledu zadané zakázky a budoucí práce celkový úsek písně selektovat a případně označit z pohledu práv budoucích uživatelů jako píseň. Proto je v tomto případě nutno brát zřetel ke kontextu celého úryvku a v tomto případě označit operní zpěv, nebo lidovou píseň nejpozději na konci předání slova moderátorem a zaměření se na píseň.

Na stanici ČRo Vltava v r. 2020 9.ledna v 11:10:10 (*LV200109-18*), signál anotován jako neřeč (jedná se z pohledu zadání o píseň).

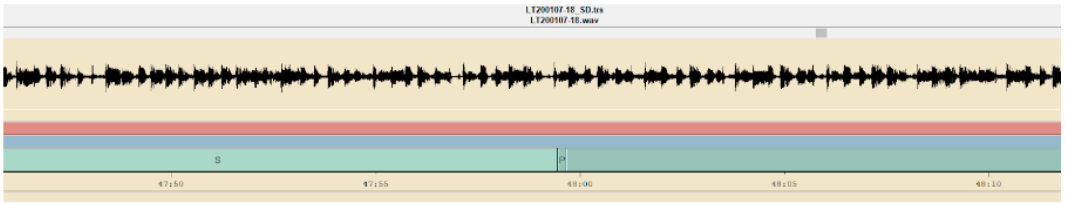


Na stanici ČRo Brno v r. 2017 14.října ve 12:55:40 *LB171014-12* je příklad Dechové lidové hudby se zpěvkyní (takovýto úsek je z pohledu zadání neřeč).



Umělé Znělky řeči

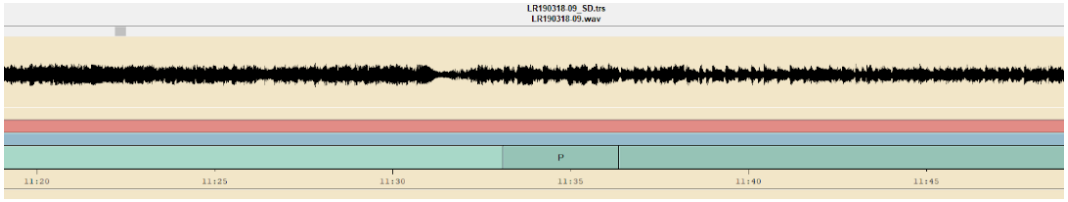
Na stanici Radio Wave, je vybrán pořad Modeshow, jako příklad LT200107-18, kde se velmi těžce i kontextově detekuje řeč, znělky s řečí, znělky bez řeči, tedy šum nebo reklamy a samotné písně. Tento konkrétní úryvek je zde zobrazen proto, že i detektor označil část písně za řeč. Dle tvaru signálu na obrázku se jedná o řečové signály, při poslechu však zjišťujeme, že se jedná o remixovatelné prvky, které svým obsahem anotátorovi přijdou spíše již jako vsunutá píseň. Největším problémem jsou pak na celé této stanici umělé znělky. Je však nutné učit potencionální nové detektory, že znělky na této stanici jsou řečové signály a je možné je i rozpoznat. Těmito znělkami totiž obvykle končí vstup moderátorů a začíná konkrétní píseň. Mnohdy plynulý přechod z remixované znělky skládající se ze slova *(*"ráááádio wavewawawaveeeee - děláme vlny"), pak končí často na konci této znělky, byť je již znělka málo slyšet. Systém se pak takovéto konce dokáže naučit a dokáže detekovat jakýsi přechod byť i klidně z mnohdy potlačené znělky do jiného úryvku. Je-li za touto znělkou hlas moderátora, samozřejmě je snaha anotování jako řečového signálu.



Ze stanice Regína pak přidáváme další znělku, utlumený a ozvěnový šepot. Znělce se dá rozumět a obvykle ohraničuje začátek pořadů. Bohužel, pro takovéto anotování je potřeba jak kontextu, tak i praxe anotování několika

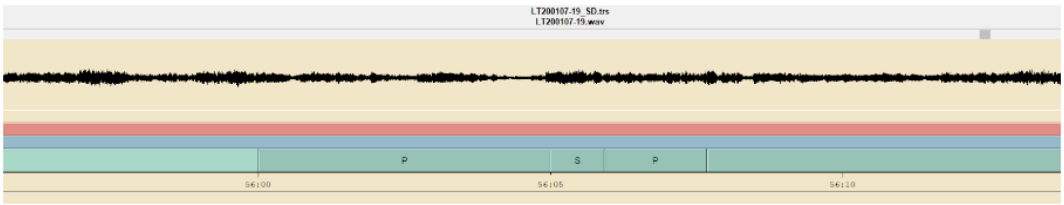
jednotek desítek od anotovaných hodin. Úryvek je z r. 2017 18.března 09:11:35 ze souboru *LR170318-09*

Umělé Znělky řeči



Málo známé a pomalé písně

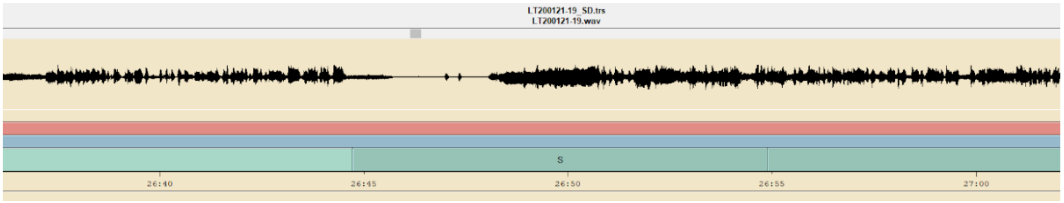
V tomto úryvku je uveden příklad pomalejší písně. Konkrétně tento úsek byl anotován na stanici Radia Wave a pochází z r. 2020 7.ledna, zejména úseky okolo 19:56:00 a okolo 19:58:00 píseň Anges Obdel ilend of doom. Zde již detektor při předzpracování vykazuje na výstupu velkou nejistotu, jedná-li se o řeč či neřeč. Vzhledem k tomu, že zní v této a podobných písních (z českých písní například Tomáš Klus v souboru LB171014-06 00:48:48) na začátku, či konci, ale někdy i uprostřed, čistý lidský hlas (nebo spíše i v těchto případech řeč) bez doprovodu hudebních nástrojů, je takovýto úsek těžko detekovatelný jako neřeč. Ještě obtížnější je, když je originál písně doplněn o před scénku - např. Tomáš Klus, píseň: *"panubohudooken"*. Detektor pak musí být trénován na těchto konkrétních problémových úsecích s dostatečně velkým kontextem. V tomto konkrétním případě je totiž možný i fakt, že následně se nezachytí hlas moderátora, jehož hlas je kodován do velké blízkosti právě slovům patřících však ještě do písně.



Smích moderátora

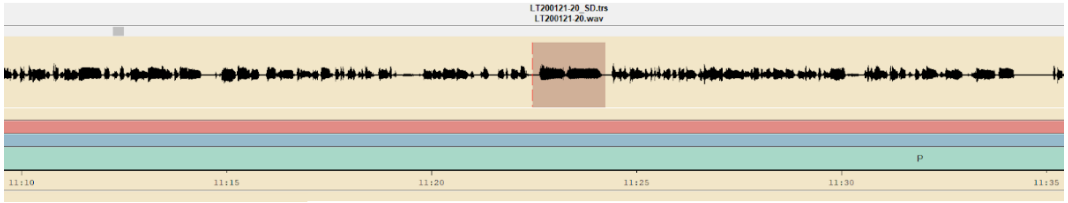
Do jisté míry rušivým problémem bývají dlouhé neřečové průniky moderátorů, hostů, nebo přispívajících dalších osob do vysílání. Specifické jsou situace

např. o Silvestru, kdy je po dobu několika minut jen slyšet jásot, či smích, tleskání a nádechy. Proto je nutné tyto části za určitých pravidel selektovat. Např. selektovat smích trvající alespoň 5 vteřin. Jestliže je takovýto neřečový, ale lidský vstup následován neřečí, je zahrnut tento takovýto úsek do celkového neřečového bloku. Stejně tomu tak je, je-li konec neřečového bloku ale ve studiu se nemluví, ale právě je slyšet smích beze slov, nebo jakýkoli jiný audio zvuk, neobsahující informaci slova. Příklad je úsek ze stanice Radio Wave LT200121-19 tedy r. 2020, 21.ledna v 19:26:45



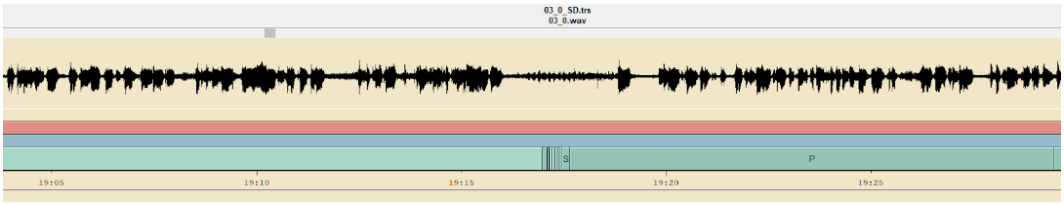
Dlouhé předěly v rozmluvě

Také dlouhé pomlky a různé výdechy mohou působit komplikace. Mezi těmito mezerami je nutné nastavit jednotný postup, jsou místa kde po telefonickém vytáčení se dlouho nikdo neozývá, nebo je slyšet jen hluk a šum. Tyto úseky by měly být vyhodnoceny jako neřečové signály. Podobně jako host ve studiu, který několik vteřin přemýšlí nad odpovědí a pak nejistě vydá zvuk "ehm"a je opět přerušen moderátorem na doplnění otázky. Je-li však pomlka krátká, tedy do 5ti vteřin a skutečně host navazuje souvislou řečí, jsou úseky ponechány jako řeč.



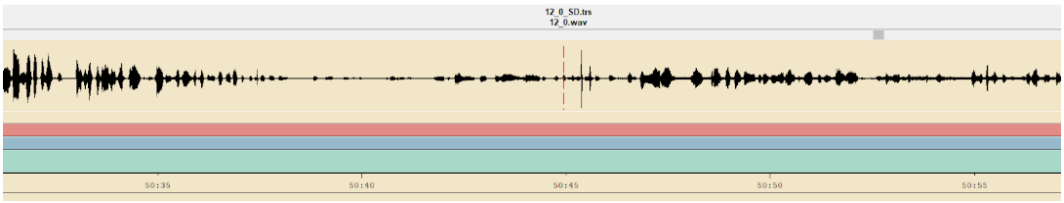
Příspěvky z terénu - parazitní hluk

Parazitní hluk, např. posolení pokrmu nebo zvuk motoru, jsou určovány jako neřeč neřeč za předpokladu, že na pozadí není žádná řeč, který by mohla být dekodována a úsek má před začátkem a po konci samotného hluku alespoň jednu celou vteřinu.



Potlačené nebo ztišené telefonáty

Není výjimkou, že se v průběhu vysílání stane technická chyba, např. v příspěvcích, kdy je do studia veden telefonát, může být hlas ztišen, zašuměn nebo se ztrácí. Nejobtížnější jsou situace, kdy nefunkční telefonát není ukončen. Následné parazitní signály ve formě šumů, praskání, ozvěn, je potřebné selektovat. V následující ukázce je šepot telefonátu, kdy je však ještě stále možno rozpoznat informaci a proto je tento úryvek vyznačen jako řečový signál.



Na úplný závěr této přílohy předkládám čtenáři pro posouzení, jak by samostatně rozhodl o řeči/neřeči ze stanice ČRo Brno.

12.listopadu 2017 čas 01:26:15. Jedná se o ukázku z divadla komik, hraného hercem Lábusem a Kaisserem, kde jejich výkony projevů zvířat jsou neuvěřitelné, nicméně pro voice activity detektor velmi složité. Naprosto plynulý přechod z řeči do kvokání (téměř maximální shoda se zvukem kura domácího), není lehký ani pro anotátora samotného.

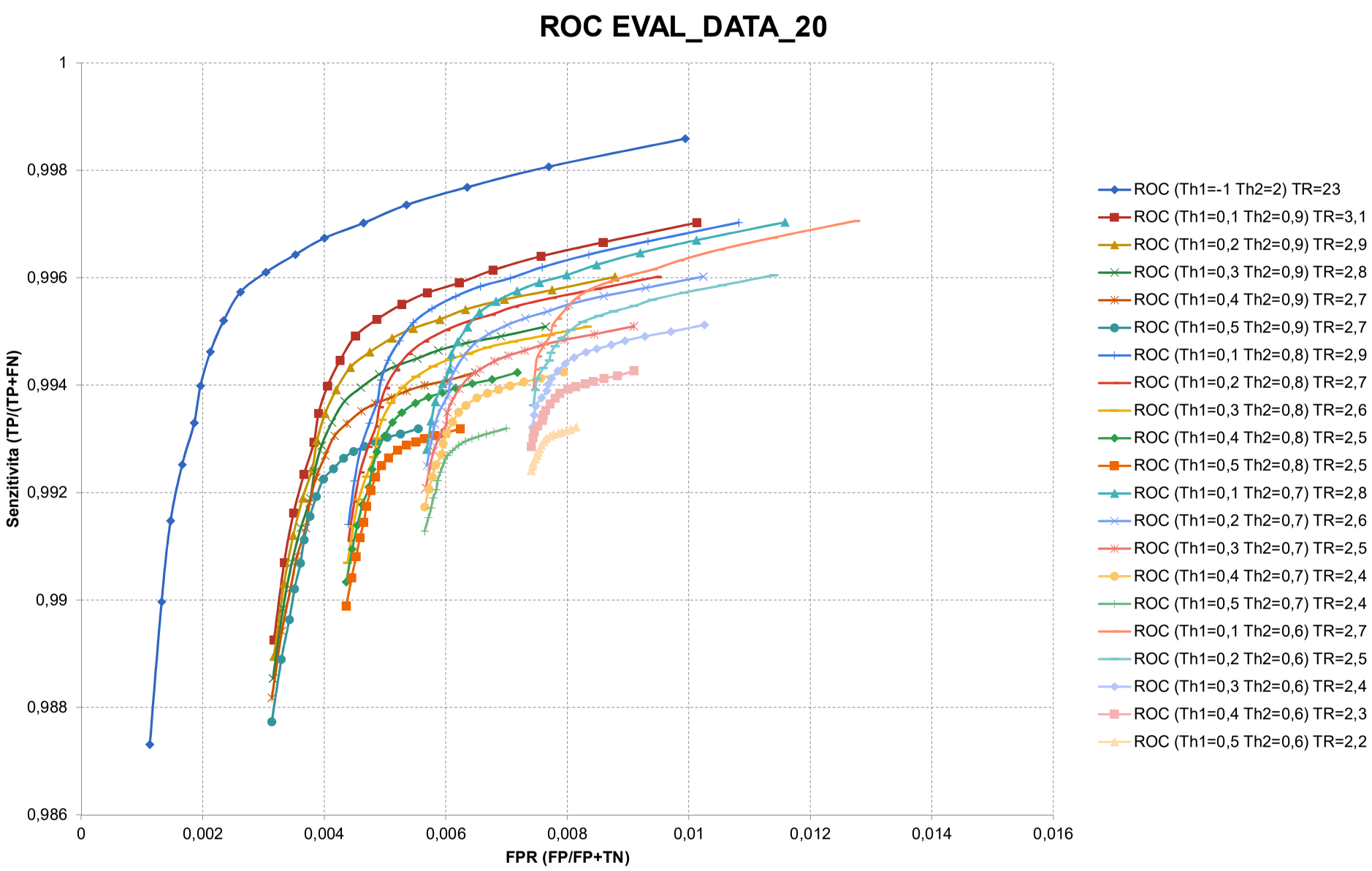
12.listopadu 2017 v čase 09:20:00. Zde se zase jedná o vyprávění pohádky s dlouhými pomlkami. LB171112-11 cca 32:30 Ptáci - pěkné zpívání Pěnice obecné Moravské.

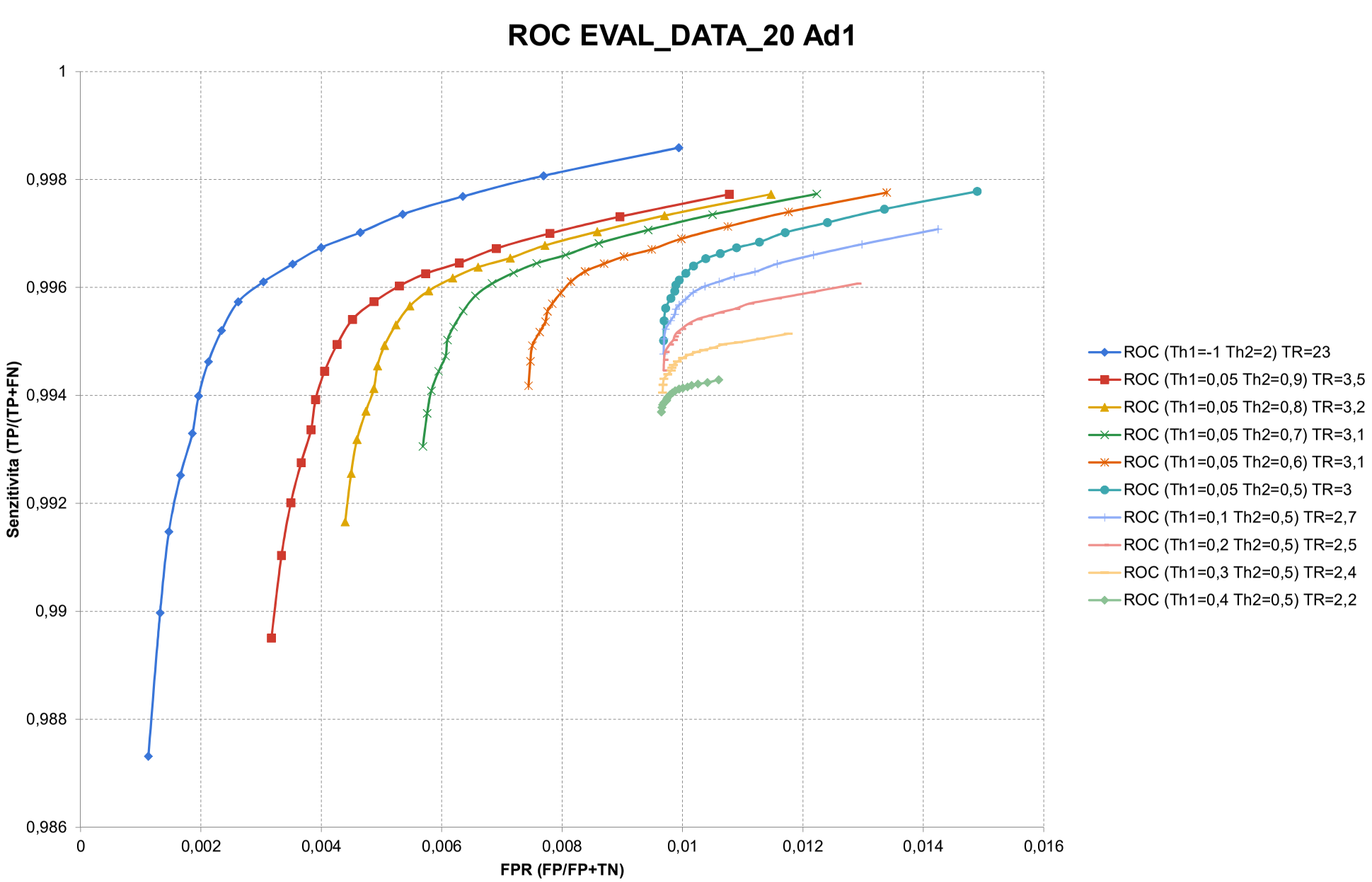
Příloha B

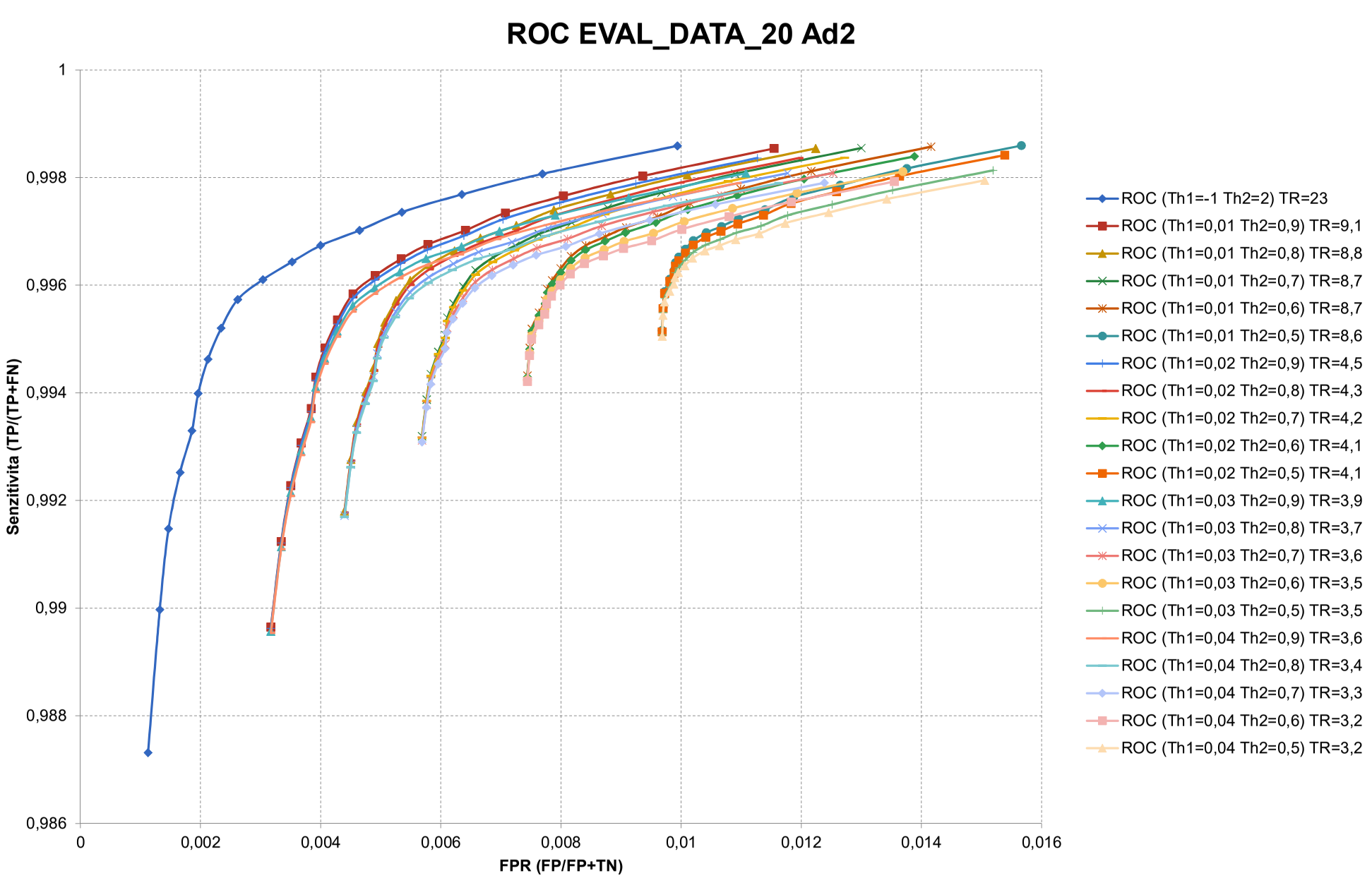
V této příloze je grafické znázornění ostatních provedených testů pro různé datasety a různé nastavení vah. Dataset označený písmenem Z byl zpracován externími anotátory.

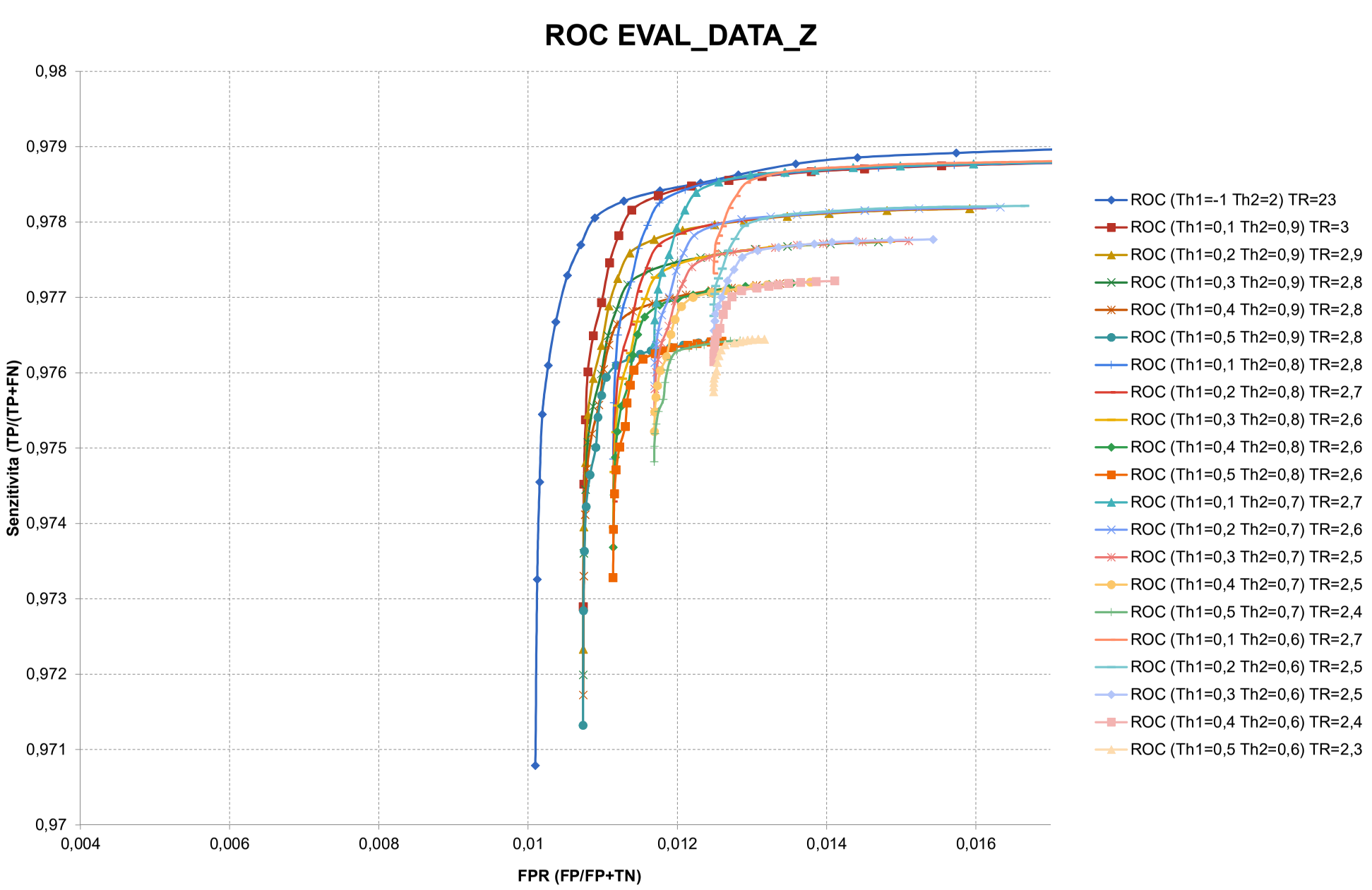
Legenda ke grafům:

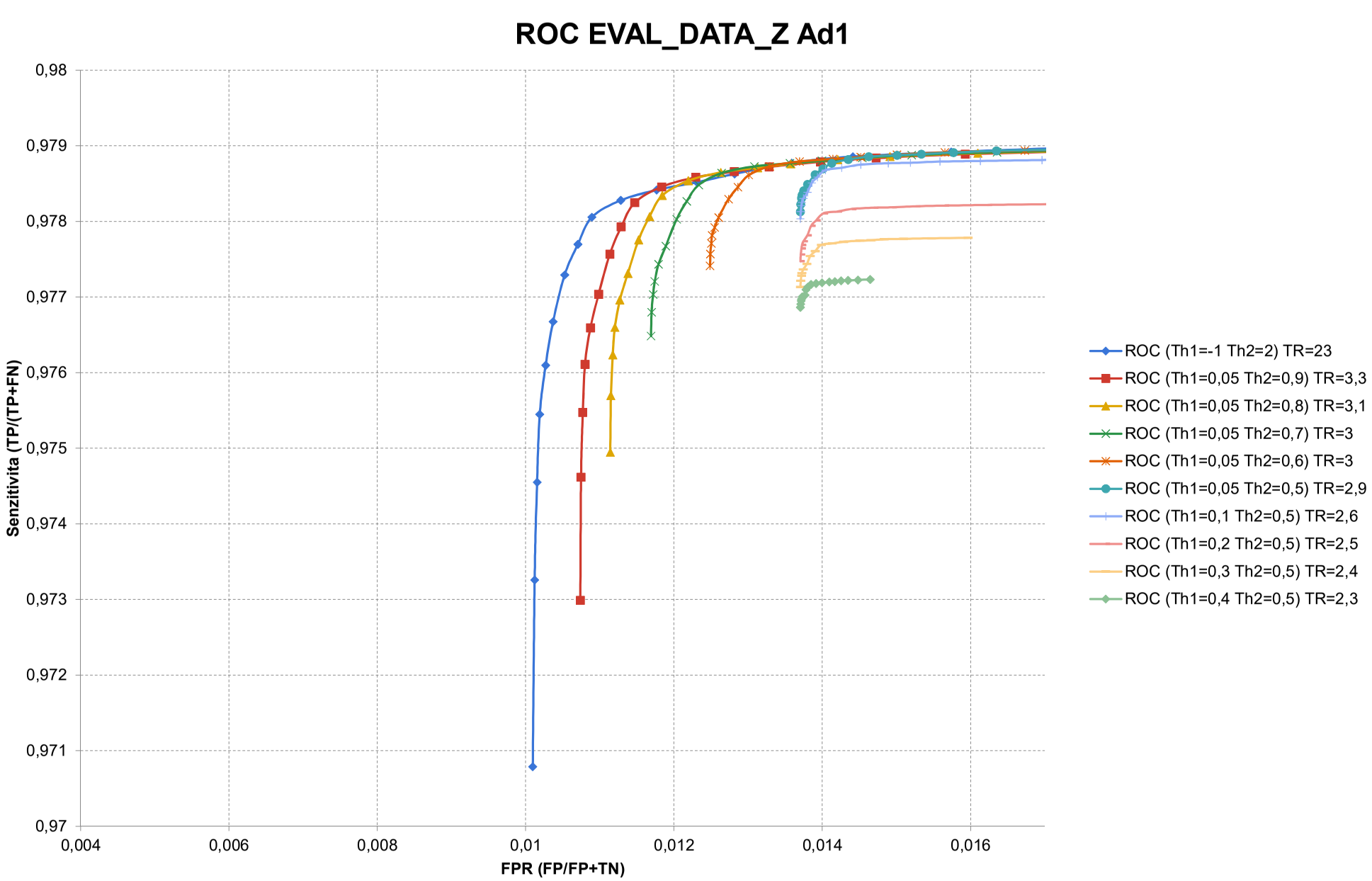
Z pohledu při čtení na šířku stránky je z pohledu čtenáře nad zobrazením ROC křivek název obsahující druh datasetu a případně jeho velikost v hodinách. Písmenem Z je označen dataset anotátorů. Tento dataset čítal 20´anotovaných hodin. Vpravo je pak popis jednotlivých realizací zpracovávanou váhou. Na osách y se vynáší sensitivita, na ose x se vynáší specificita.

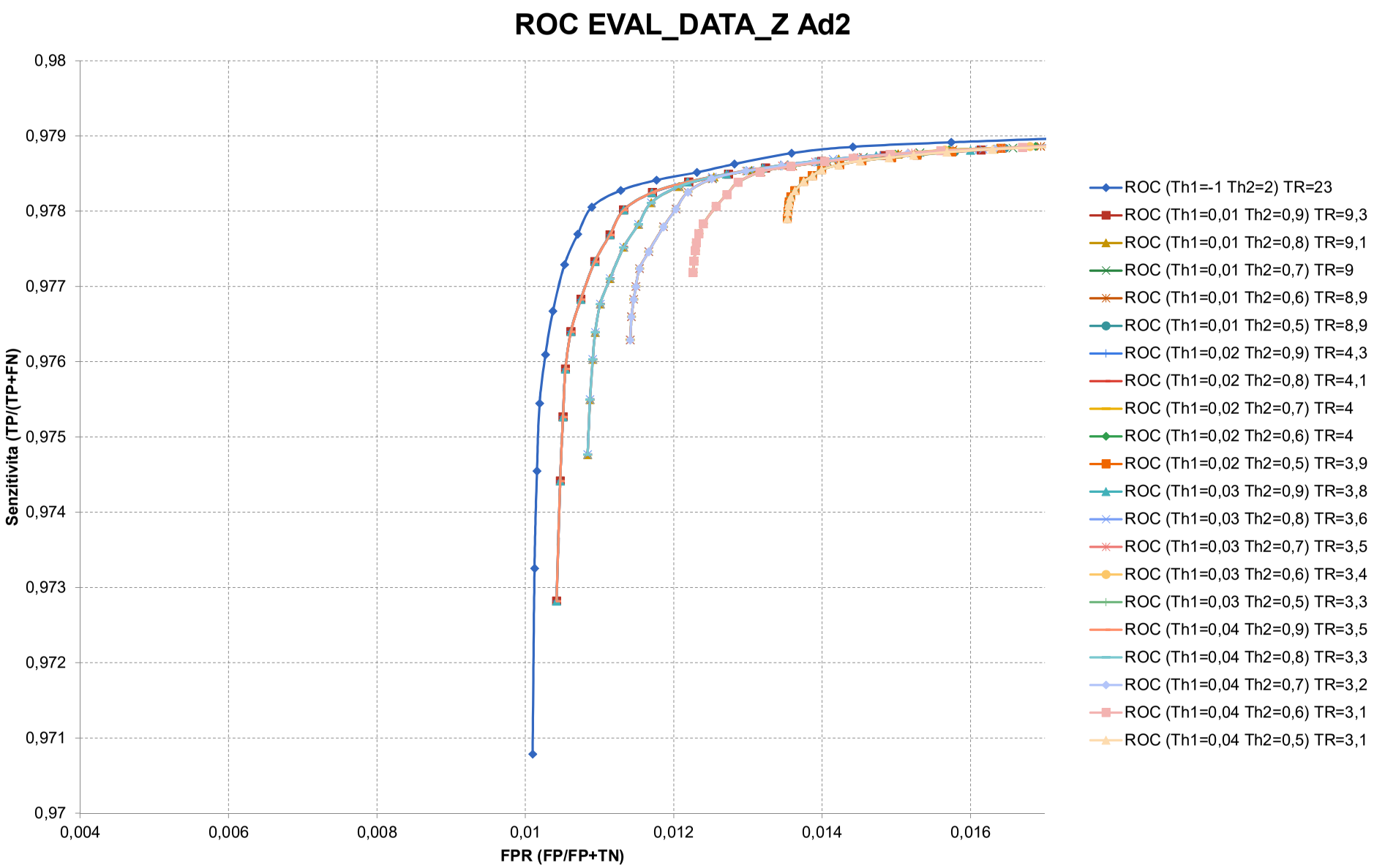

ROC EVAL_DATA_20
Senzitivita (TP/(TP+FN)
FPR (FP/FP+TN)
1
0,998
0,996
0,994
0,992
0,99
0,988
0,986
0
0,002
0,004
0,006
0,008
0,01
0,012
0,014
0,016
ROC (Th1=-1 Th2=2) TR=23
ROC (Th1=0,1 Th2=0,9) TR=3,1
ROC (Th1=0,2 Th2=0,9) TR=2,9
ROC (Th1=0,3 Th2=0,9) TR=2,8
ROC (Th1=0,4 Th2=0,9) TR=2,7
ROC (Th1=0,5 Th2=0,9) TR=2,7
ROC (Th1=0,1 Th2=0,8) TR=2,9
ROC (Th1=0,2 Th2=0,8) TR=2,7
ROC (Th1=0,3 Th2=0,8) TR=2,6
ROC (Th1=0,4 Th2=0,8) TR=2,5
ROC (Th1=0,5 Th2=0,8) TR=2,5
ROC (Th1=0,1 Th2=0,7) TR=2,8
ROC (Th1=0,2 Th2=0,7) TR=2,6
ROC (Th1=0,3 Th2=0,7) TR=2,5
ROC (Th1=0,4 Th2=0,7) TR=2,4
ROC (Th1=0,5 Th2=0,7) TR=2,4
ROC (Th1=0,1 Th2=0,6) TR=2,7
ROC (Th1=0,2 Th2=0,6) TR=2,5
ROC (Th1=0,3 Th2=0,6) TR=2,4
ROC (Th1=0,4 Th2=0,6) TR=2,3
ROC (Th1=0,5 Th2=0,6) TR=2,2


ROC EVAL_DATA_20 Ad1
Senzitivita (TP/(TP+FN)
FPR (FP/FP+TN)
1
0,998
0,996
0,994
0,992
0,99
0,988
0,986
0
0,002
0,004
0,006
0,008
0,01
0,012
0,014
0,016
ROC (Th1=-1 Th2=2) TR=23
ROC (Th1=0,05 Th2=0,9) TR=3,5
ROC (Th1=0,05 Th2=0,8) TR=3,2
ROC (Th1=0,05 Th2=0,7) TR=3,1
ROC (Th1=0,05 Th2=0,6) TR=3,1
ROC (Th1=0,05 Th2=0,5) TR=3
ROC (Th1=0,1 Th2=0,5) TR=2,7
ROC (Th1=0,2 Th2=0,5) TR=2,5
ROC (Th1=0,3 Th2=0,5) TR=2,4
ROC (Th1=0,4 Th2=0,5) TR=2,2


ROC EVAL_DATA_20 Ad2
Senzitivita (TP/(TP+FN)
FPR (FP/FP+TN)
1
0,998
0,996
0,994
0,992
0,99
0,988
0,986
0
0,002
0,004
0,006
0,008
0,01
0,012
0,014
0,016
ROC (Th1=-1 Th2=2) TR=23
ROC (Th1=0,01 Th2=0,9) TR=9,1
ROC (Th1=0,01 Th2=0,8) TR=8,8
ROC (Th1=0,01 Th2=0,7) TR=8,7
ROC (Th1=0,01 Th2=0,6) TR=8,7
ROC (Th1=0,01 Th2=0,5) TR=8,6
ROC (Th1=0,02 Th2=0,9) TR=4,5
ROC (Th1=0,02 Th2=0,8) TR=4,3
ROC (Th1=0,02 Th2=0,7) TR=4,2
ROC (Th1=0,02 Th2=0,6) TR=4,1
ROC (Th1=0,02 Th2=0,5) TR=4,1
ROC (Th1=0,03 Th2=0,9) TR=3,9
ROC (Th1=0,03 Th2=0,8) TR=3,7
ROC (Th1=0,03 Th2=0,7) TR=3,6
ROC (Th1=0,03 Th2=0,6) TR=3,5
ROC (Th1=0,03 Th2=0,5) TR=3,5
ROC (Th1=0,04 Th2=0,9) TR=3,6
ROC (Th1=0,04 Th2=0,8) TR=3,4
ROC (Th1=0,04 Th2=0,7) TR=3,3
ROC (Th1=0,04 Th2=0,6) TR=3,2
ROC (Th1=0,04 Th2=0,5) TR=3,2


ROC EVAL_DATA_Z
Senzitivita (TP/(TP+FN)
0,98
0,979
0,978
0,977
0,976
0,975
0,974
0,973
0,972
0,971
0,97
0,004
0,006
0,008
0,01
0,012
0,014
0,016
FPR (FP/FP+TN)
ROC (Th1=-1 Th2=2) TR=23
ROC (Th1=0,1 Th2=0,9) TR=3
ROC (Th1=0,2 Th2=0,9) TR=2,9
ROC (Th1=0,3 Th2=0,9) TR=2,8
ROC (Th1=0,4 Th2=0,9) TR=2,8
ROC (Th1=0,5 Th2=0,9) TR=2,8
ROC (Th1=0,1 Th2=0,8) TR=2,8
ROC (Th1=0,2 Th2=0,8) TR=2,7
ROC (Th1=0,3 Th2=0,8) TR=2,6
ROC (Th1=0,4 Th2=0,8) TR=2,6
ROC (Th1=0,5 Th2=0,8) TR=2,6
ROC (Th1=0,1 Th2=0,7) TR=2,7
ROC (Th1=0,2 Th2=0,7) TR=2,6
ROC (Th1=0,3 Th2=0,7) TR=2,5
ROC (Th1=0,4 Th2=0,7) TR=2,5
ROC (Th1=0,5 Th2=0,7) TR=2,4
ROC (Th1=0,1 Th2=0,6) TR=2,7
ROC (Th1=0,2 Th2=0,6) TR=2,5
ROC (Th1=0,3 Th2=0,6) TR=2,5
ROC (Th1=0,4 Th2=0,6) TR=2,4
ROC (Th1=0,5 Th2=0,6) TR=2,3


ROC EVAL_DATA_Z Ad1
Senzitivita (TP/(TP+FN)
0,98
0,979
0,978
0,977
0,976
0,975
0,974
0,973
0,972
0,971
0,97
0,004
0,006
0,008
0,01
0,012
0,014
0,016
FPR (FP/FP+TN)
ROC (Th1=-1 Th2=2) TR=23
ROC (Th1=0,05 Th2=0,9) TR=3,3
ROC (Th1=0,05 Th2=0,8) TR=3,1
ROC (Th1=0,05 Th2=0,7) TR=3
ROC (Th1=0,05 Th2=0,6) TR=3
ROC (Th1=0,05 Th2=0,5) TR=2,9
ROC (Th1=0,1 Th2=0,5) TR=2,6
ROC (Th1=0,2 Th2=0,5) TR=2,5
ROC (Th1=0,3 Th2=0,5) TR=2,4
ROC (Th1=0,4 Th2=0,5) TR=2,3


ROC EVAL_DATA_Z Ad2
Senzitivita (TP/(TP+FN)
FPR (FP/FP+TN)
0,98
0,979
0,978
0,977
0,976
0,975
0,974
0,973
0,972
0,971
0,97
0,004
0,006
0,008
0,01
0,012
0,014
0,016
ROC (Th1=-1 Th2=2) TR=23
ROC (Th1=0,01 Th2=0,9) TR=9,3
ROC (Th1=0,01 Th2=0,8) TR=9,1
ROC (Th1=0,01 Th2=0,7) TR=9
ROC (Th1=0,01 Th2=0,6) TR=8,9
ROC (Th1=0,01 Th2=0,5) TR=8,9
ROC (Th1=0,02 Th2=0,9) TR=4,3
ROC (Th1=0,02 Th2=0,8) TR=4,1
ROC (Th1=0,02 Th2=0,7) TR=4
ROC (Th1=0,02 Th2=0,6) TR=4
ROC (Th1=0,02 Th2=0,5) TR=3,9
ROC (Th1=0,03 Th2=0,9) TR=3,8
ROC (Th1=0,03 Th2=0,8) TR=3,6
ROC (Th1=0,03 Th2=0,7) TR=3,5
ROC (Th1=0,03 Th2=0,6) TR=3,4
ROC (Th1=0,03 Th2=0,5) TR=3,3
ROC (Th1=0,04 Th2=0,9) TR=3,5
ROC (Th1=0,04 Th2=0,8) TR=3,3
ROC (Th1=0,04 Th2=0,7) TR=3,2
ROC (Th1=0,04 Th2=0,6) TR=3,1
ROC (Th1=0,04 Th2=0,5) TR=3,1

Příloha C

Zde je uveden konfigurační skript modelu BASE, užitého v této práci. Tento skript vychází z otevřeného zdrojového kodu, který definuje strukturu modelu transformeru na hugging face portálu.

"_name_or_path": "***/Wav2Vec2_pretrained_cz",
"activation_dropout": 0.1,
"apply_spec_augment": true,
"architectures": [
"Wav2Vec2ForTokenClassification"
],
"attention_dropout": 0.1,
"bos_token_id": 1,
"classifier_proj_size": 256,
"codevector_dim": 256,
"contrastive_logits_temperature": 0.1,
"conv_bias": false,
"conv_dim": [
512,
512,
512,
512,
512,
512,
512
],
"conv_kernel": [
10,
3,
3,
3,
3,
2,

```
2 ],
"conv_stride": [
5,
2,
2,
2,
2,
2,
2
],
"ctc_loss_reduction": "sum",
"ctc_zero_infinity": false,
"diversity_loss_weight": 0.1,
"do_stable_layer_norm": false,
"eos_token_id": 2,
"feat_extract_activation": "gelu",
"feat_extract_norm": "group",
"feat_proj_dropout": 0.1,
"feat_quantizer_dropout": 0.0,
"final_dropout": 0.1,
"gradient_checkpointing": false,
"hidden_act": "gelu",
"hidden_dropout": 0.1,
"hidden_size": 768,
"initializer_range": 0.02,
"intermediate_size": 3072,
"layer_norm_eps": 1e-05,
"layerdrop": 0.1,
"mask_feature_length": 10,
"mask_feature_prob": 0.0,
"mask_time_length": 10,
"mask_time_prob": 0.05,
```

"model_type": "wav2vec2",
"num_attention_heads": 12,
"num_codevector_groups": 2,
"num_codevectors_per_group": 320,
"num_conv_pos_embedding_groups": 16,
"num_conv_pos_embeddings": 128,
"num_feat_extract_layers": 7,
"num_hidden_layers": 12,
"num_negatives": 100,
"pad_token_id": 0,
"proj_codevector_dim": 256,
"torch_dtype": "float32",
"transformers_version": "4.10.3",
"use_weighted_layer_sum": false,
"vocab_size": 32

Zde následuje konfigurační nastavení modelu SMALL užitého v této práci.

"_name_or_path": "***/Wav2Vec2_tiny25Hz-v0.2_pretrained_cz"
"activation_dropout": 0.1
"apply_spec_augment": true
"architectures": [
"Wav2Vec2ForTokenClassification"]
"attention_dropout": 0.1
"bos_token_id": 1
"classifier_proj_size": 16
"codevector_dim": 16
"contrastive_logits_temperature": 0.1
"conv_bias": false
"conv_dim": [
512
512
512
512
512
512
512
512
]
"conv_kernel": [
10
3
3
3
3
2
2
2
]

"conv_stride": [

5

2

2

2

2

2

2

2

]

"ctc_loss_reduction": "sum"

"ctc_zero_infinity": false

"diversity_loss_weight": 0.1

"do_stable_layer_norm": false

"eos_token_id": 2

"feat_extract_activation": "gelu"

"feat_extract_norm": "group"

"feat_proj_dropout": 0.1

"feat_quantizer_dropout": 0.0

"final_dropout": 0.1

"gradient_checkpointing": false

"hidden_act": "gelu"

"hidden_dropout": 0.1

"hidden_size": 384

"initializer_range": 0.02

"intermediate_size": 1536

"layer_norm_eps": 1e-05

"layerdrop": 0.1

"mask_feature_length": 5

"mask_feature_prob": 0.0

"mask_time_length": 5

"mask_time_prob": 0.05

"model_type": "wav2vec2"
"num_attention_heads": 6
"num_codevector_groups": 2
"num_codevectors_per_group": 320
"num_conv_pos_embedding_groups": 16
"num_conv_pos_embeddings": 64
"num_feat_extract_layers": 8
"num_hidden_layers": 3
"num_negatives": 100
"pad_token_id": 0
"proj_codevector_dim": 16
"torch_dtype": "float32"
"transformers_version": "4.11.3"
"use_weighted_layer_sum": false
"vocab_size": 32